

YOUR **COMMODORE** £1.50  
**SERIOUS  
USERS  
GUIDE** **1987**

THE ULTIMATE  
GUIDE FOR  
COMMODORE  
OWNERS

INCLUDING:

A SUPERB C64 80 COLUMN  
WORDPROCESSOR

PLUS/4 EXTENDED BASIC

PROTECTING YOUR  
PROGRAMS FROM  
PRYING EYES

NEW CHARACTER  
SETS FOR YOUR  
MPS801/3

TECHNICAL  
INFORMATION  
FOR THE C64,  
C128, PLUS/4  
AND C16



FROM THE PUBLISHERS OF YOUR COMMODORE



# THE PARALLEL DISK TURBO SYSTEM EVERY 1541 WANTS!

## THE NEW! PHANTOM

NOW WITH ADDED POWER!

- LOADS 240 BLOCKS IN 7secs
- LOADS 84 BLOCKS IN 2secs
- SAVES 84 BLOCKS IN 4secs
- FORMAT 35 TRACKS IN 20secs
- FORMAT 40 TRACKS IN 23secs

screen on, with  
FULL ERROR CHECKING

with TRACK by TRACK VERIFY

- EXPERT COMPATIBLE – PROGRAM THE EXPERT IN LESS THAN 2secs.
- SPEEDS UP ALL DRIVE FUNCTIONS

- FULL ERROR CHECKING RETAINED – ESSENTIAL FOR RELIABILITY  
Other systems sacrifice this CRUCIAL FUNCTION to achieve speed increases but are in fact, NO FASTER only LESS RELIABLE.  
Buy their systems and you'll find out the hard way!

- 8 FUNCTION KEY COMMANDS FOR MAJOR FUNCTIONS.

- 60 ADDITIONAL COMMANDS INCLUDING:

- FILE LOCK & UNLOCK

- WRITE PROTECT IGNORE – NO NEED TO NOTCH DISKS

- SET DEVICE NUMBER ● SCREEN ON/OFF

- GO1541 – REVERTS TO STANDARD 1541

- MANY OTHER USEFUL COMMANDS AID COMPATIBILITY.

(Disable function keys, disable extra ram etc.)

- DRIVE MONITOR – COMMANDS INCLUDE: DISASSEMBLE, FILL, COMPARE, HUNT, ASSEMBLE, EXECUTE ETC. ETC.

- COMPREHENSIVE 40 PAGE MANUAL & ILLUSTRATED FITTING GUIDE.

- COMPATIBLE WITH MOST COMMERCIAL SOFTWARE.

- RELIABLE – DOES NOT CORRUPT DISKS UNLIKE RIVAL PRODUCTS.

- SWITCHABLE KERNAL REPLACEMENT INCLUDED FOR 64 OR 128 (64MODE).

- BRITISH DESIGNED AND MANUFACTURED - AVAILABLE EXCLUSIVELY FROM TRILOGIC.

- UPGRADABLE – DOS, KERNAL & COPIER UPGRADES WILL BE AVAILABLE SOON TO GIVE EXTRA FEATURES FOR A NOMINAL SUM.

### OPTIONAL EXTRAS

#### DRIVE COOLING KIT 1.

RAISES THE LID BY 1.5" TO  
ENABLE GREATER AIR CIRCULATION

PRICE

£2.50

#### DRIVE COOLING KIT 2.

Consists of a very quiet mains operated  
80mm fan unit which rests on the drive  
lid over the ventilation slots

PRICE

£17.95

## THE EXPERT CARTRIDGE WITH BUILT IN E.S.M.!

- SAVES PROGRAMS IN ONE FILE – (xcl. multipart ones)
- COMPACTS PROGRAMS used by leading software houses.
- RELOADS, DECOMPACTS & RUNS ANY PROGRAM WITHIN 35 secs.
- THE EXPERT IS NOT NEEDED FOR RELOADING.
- COMPATIBLE WITH THE PHANTOM & ALL CBM64/128 DISK DRIVES.

WITH THE NEW V2.10 SOFTWARE SUPPLIED,  
THE EXPERT CAN DO ALL THIS AND MORE.

### PROGRAMME PARALYSER

Stops even the most heavily protected programs  
and defeats All "anti-freeze" techniques.

### BACK-UP GENERATOR

TAPE/DISK, ● DISK/TAPE, ● DISK/DISK,  
● TAPE/TAPE Transfers memory resident  
programs whether loaded from disk or tape.

### SPRITE EXTRACTOR & IMMORTALISER



#### HIRES SCREEN GRABBER

#### CHEAT MACHINE

#### CODE INTERROGATOR

#### THE ONLY PROGRAMMABLE CARTRIDGE



### FREE DISK COPIER

WORTH £20.00

### WITH EVERY PHANTOM

This copier uses the Phantom  
hardware to recreate bit by bit,  
a duplicate of hitherto  
uncopyable disks.

### PRICE

£78.99

inc VAT  
& p&p

### PRICE

£29.99

inc VAT  
& p&p

EXPRESS  
DELIVERY  
ADD £1.95

FAST MAIL ORDER SERVICE ● PROMPT DESPATCH ● ALL PRICES FULLY INCLUSIVE PLUS 10 DAY MONEY BACK  
GUARANTEE ON ALL BACK-UP DEVICES. ORDERING: WRITE OR 'PHONE PAYMENT BY CASH, CHEQUES PAYABLE  
TO TRILOGIC/ BY POSTAL ORDER OR ACCESS ● ADD £1 EXTRA FOR EXPORT ORDERS. PAYMENT IN STERLING ONLY PLEASE.

DEPT PX 3 329 TONG STREET BRADFORD BD4 9QY TEL (0274) 691115



# YOUR COMMODORE SERIOUS USERS GUIDE 1987

Editor: Stuart Cooke  
Assistant Editor: Sue Joyce  
Editorial Assistant: Kirk Rutter  
Senior Advertising Manager: Pete Chandler  
Advertisement Manager: Stuart Taylor  
Advertisement Copy Control: Laura Champion  
Typesetting: Project 3  
Design: ASP Art Studio

Argus Specialist Publications Limited Editorial & Advertisement Office, Your Commodore, No 1 Golden Square, London W1R 3AB.  
Telephone: 01-437 0626. Telex: 8811896.

## CONTENTS

<b>LISTINGS</b> <b>4</b>	<b>CHARACTER SCROLLER</b> <b>18</b>	<b>NEW CHARACTERS ON THE</b>
How to type in the programme in this magazine.	Scrolling character sets for C64 owners.	<b>MPS 801/3</b> <b>49</b>
<b>HASHING IT WITH CBM</b> <b>6</b>	<b>TRANS-SCRIPT</b> <b>21</b>	Give your MPS801/3 a character set of your own design.
Using relative files with your disk drive.	Convert your Plus/4 3plus1 files to Script Plus.	<b>YC WRITER</b> <b>56</b>
<b>FAST FORMATTER</b> <b>8</b>	<b>PLUS/4 EXTENDED BASIC</b> <b>23</b>	A superb 80 column wordprocessor for C64 owners.
Format a disk in under 10 seconds (C64).	A host of new commands for Plus/4 owners.	<b>TECHNICAL INFORMATION</b> <b>71</b>
<b>MULTIFILE</b> <b>10</b>	<b>WORDPROCESSOR ROUND UP</b> <b>30</b>	All you ever wanted to know about your computer.
Customize this C64 database to suit your own requirements.	What's the best wordprocessor for the money?	<b>FOREIGN FORMATS</b> <b>85</b>
<b>DISK FILE DESCRIBER</b> <b>16</b>	<b>EVERY MANS GUIDE TO</b>	Using your 1571 to read strange disks.
Keep track of the programmes on your C64 disks.	<b>GRAPHICS</b> <b>34</b>	<b>PRINT MASTER</b> <b>87</b>
	Everything you wanted to know about C64 graphics.	Produce Sprite, character and screen grids with this handy C64 program.
	<b>SWAPPER 64</b> <b>41</b>	<b>PROGRAM LOCK</b> <b>89</b>
	Swap between two programmes at the press of a key.	Keep prying eyes out of your latest programming masterpiece.
	<b>128 DISK UTILITY</b> <b>43</b>	<b>SOFTWARE FOR SALE</b> <b>90</b>
	A utility no C128 owner should be without.	How to buy these programmes on disk or cassettes.

The Your Commodore Serious User Guide is packed full of vital information and programmes for all types of Commodore owners.

If you use your computer for 'home office' purposes then the 80 column C64 wordprocessor will no doubt come in extremely handy (tape and disk supported). If you need to keep lists of information, the database Multifile will be very useful. Multifile is customized to suit your specific needs and it can be used for anything from running a stock control to keeping a list of names and addresses.

Many utility programmes are also included. MPS801/3 owners

can now use descenders and user-defined character sets on their printers. Plus/4 owners can add a wealth of new commands with our extended Basic. Disk owners will find the fast formatter and file describer invaluable utilities.

If you write your own programmes then there's plenty in the Guide for you too. Program Lock will protect your programmes and keep prying eyes from reading your code. A character scroller for the C64 will help to improve the visual effect of your programs.

Beginners and hardened programmers alike will find the

wealth of technical information provided in our Technical Appendix, an invaluable reference. Here you will find memory maps for all of the popular Commodore computers. ROM calls are also listed so that you can find out, at a glance, information that you need when programming. Aids such as the hex-decimal convertor and the list of useful POKE commands will also prove extremely useful.

The Your Commodore Serious Users Guide is something that no Commodore owner should be without.



# Listings

*Get it right first time with our deluxe program system  
for the C64.*

**Y**ou may have noticed that our listings are free of those horrible little black blobs which send you searching around the keyboard for a suitable graphic symbol. You may also have noticed the funny numbers by the side of each line of the listing. Fret no more, it's all part of our easy entry aid.

Instead of those nasty graphics and rows of countless spaces in PRINT statements and strings we use a special coding system. The code, or mnemonic, is always contained in square brackets and you'll soon learn to decipher their meanings.

For example, [SA] would mean type in a Shifted A, or an ace of spades in layman's terms, and [SA10] would mean a row of ten of these symbols.

[S+2] means hold down the shift key and press the plus key twice. It doesn't take a great leap of logic to realise that [C+2] means exactly the same thing except that the Commodore key (bottom left of the keyboard) is held down instead of the shift key.

If more than two spaces appear in a statement then this will be printed as [SPC4] or, exceptionally, [SSPC4]. Translated into English this means press the spacebar four times or in the latter case hold the shift key down while you do it.

A string of special characters could appear as:

[CTRL N, DOWN2,LEFT5,BLUE, F3,C3]

This would be achieved by holding

down the CTRL key as you press N, press the cursor key down twice, the cursor left key five times, press the key marked BLUE while holding down the CTRL key, press the F3 key and, finally hold the Commodore key down while pressing the number two key (C2 would of course make the computer print in brown).

Always remember that you should only have a row of graphics characters on your screen with no square brackets and no commas, unless something like this appears:

[SS],[C\*]

In this case the two characters should have a comma between them.

On rare occasions [REV T] will appear in a listing. This is a delete symbol and is created by entering the line up to this mnemonic. Then type a closing quotation mark (SHIFT & 2) and delete it. This gets the computer out of quotes mode. Hold down CTRL and press the number nine key (RVSON), type the relevant number of reversed T's and then hold down CTRL and press zero (RVSOFF). Next type another quotation mark and delete it again. Now finish the line and press RETURN.

A list of these special cases is given in the table but remember that only one of these mnemonics will appear outside of a PRINT string: the symbol for pi. This may appear when its value is needed in a calculation so this may look something like:

:CC=2\*[PI]\*R:

Ignore the square brackets and just type in a shifted upward pointing arrow (ie. the pi symbol).

## PROGRAM: SYNTAX CHECKER

```




























5 REM SYNTAX CHECKER - ERIC DOYLE
10 BL=10 :LN=70 :SA=49152
20 FOR L=0 TO BL:GX=0:FOR D=0 TO
  15
30 READ A:IF A>255THENPRINT"NUMB
  ER TO LARGE":LN+(L*10):STOP
40 GX=GX+A:POKE SA+L*16+D,A:NEXT
  D
50 READ A:IF A<GX THENPRINT"ERR
  OR IN LINE":LN+(L*10):STOP
60 NEXT L:SYS 49152:NEW
70 DATA 173,5,3,201,165,208,31,1
  20,169,9,141,32,208,141,33,208,1
  847
80 DATA 169,7,141,134,2,169,13,3
  2,210,255,169,64,141,4,3,169,168
  2
90 DATA 192,141,5,3,88,96,120,16
  9,124,141,4,3,169,165,141,5,1566

100 DATA 3,169,14,141,134,2,141,
  32,208,169,6,141,33,208,88,96,15
  85
110 DATA 32,124,165,72,138,72,15
  2,72,162,0,165,20,133,254,165,21
  ,1747
120 DATA 24,101,254,133,254,189,
  0,2,240,18,69,254,133,254,232,18
  9,2346
130 DATA 0,2,240,8,24,101,254,13
  3,254,232,208,233,169,1,141,134,
  2134
140 DATA 2,165,254,74,74,74,74,3
  2,156,192,32,210,255,165,254,41,
  2054
150 DATA 15,32,156,192,32,210,25
  5,169,13,32,210,255,169,13,32,21
  0,1995
160 DATA 255,169,7,141,134,2,104
  ,168,104,170,104,96,24,105,48,20
  1,1832
170 DATA 58,16,1,96,24,105,7,96,
  0,0,0,0,0,0,0,0,403

```

by Eric Doyle



Mnemonic	Symbol	Keypress	Mnemonic	Symbol	Keypress
[RIGHT]		CRSR left/right	[BLACK]		CTRL & 1
[LEFT]		SHIFT & CRSR left/right	[WHITE]		CTRL & 2
[DOWN]		CRSR up/down	[RED]		CTRL & 3
[UP]		SHIFT & CRSR up/down	[CYAN]		CTRL & 4
[F1]		f1 key	[PURPLE]		CTRL & 5
[F2]		SHIFT & f1 key	[GREEN]		CTRL & 6
[F3]		f3 key	[BLUE]		CTRL & 7
[F4]		SHIFT & f3 key	[YELLOW]		CTRL & 8
[F5]		f5 key	[POUND]		£
[F6]		SHIFT & f5 key	[LARROW]		←
[F7]		f7 key	[UPARROW]		↑
[F8]		SHIFT & f7 key	[PI]		SHIFT & ↑
[HOME]		CLR/HOME	[INST]		SHIFT & INST/DEL
[CLR]		SHIFT & CLR/HOME	[REV T]		see text
[RVSON]		CTRL & 9	[Cletter]		CBM + letter
[RVSOFF]		CTRL & 0	[Sletter]		SHIFT + letter

## Checksum Program

The hexadecimal numbers appearing in a column to the left of the listing should not be typed in with the program. These are merely checksum values and are there to help you get each line right. Don't worry if you don't understand the hexadecimal system, as long as you can compare two characters on the screen with the corresponding two characters in the magazine you can use our line checking program.

Type in the Checksum Program, make sure that you've not made any mistakes and save it to tape or disk

immediately because it will be used with most of the present and future listings appearing in Your Commodore.

At the start of each programming session, load Checksum and run it. The screen will turn brown with yellow characters and each time you type in a line and press the RETURN key a number will appear on the screen in white. This should be the same as the corresponding value in the magazine.

If the two values don't relate to one another, you have not copied the line exactly as printed so go back and check each character carefully. When you find the error simply correct it and

press RETURN again.

If you want to turn off the checker simply type SYS49152 and the screen will return to the familiar blue colours. You can then do whatever it was you wanted to do and if this doesn't use the area where Checksum lies you can go back to it with the same SYS command.

No system is foolproof but the chances of two errors cancelling one another out are so remote that we believe our listings are more reliable than any other magazine in the world. So get typing!



# Hashing it with Commodore

*We would all like to make more use of data files but lack of clear advice as to how to index/retrieve the data often deters programmers from using relative files to their full advantage*

A problem often encountered with databases employing relative files is that of not being able to rapidly read records without specifying the DOS record number. Clearly, searching and comparing in record number sequence through an entire data file in order to find one record defeats the purpose of random access files. Imagine being able to access a record by defining the data of one or a combination of more than one field.

For example if you have a relative data file containing six fields:

SURNAME  
FIRSTNAME  
BIRTHDATE  
STREET  
TOWN  
COUNTY

You may need to find a record by specifying the SURNAME and FIRSTNAME without even knowing by what record number the data had been filed. The purpose of a good database would preclude you keeping a list of record numbers and records so it would be most unlikely that you would know the DOS record number anyway.

The solution to this problem of how to find records by specifying the data lies in the maintenance of one or more *HASH FILES*. Yes, you may say "why HASH a program that's probably already a HASH?" well hashing doesn't quite mean to make a mess of it!

One creates a number called a *hash number* by performing a mathematical calculation upon the data in defined data fields. That *hash number* (referred to hereafter as *HASH(No.)*) is ideally

unique to any set of data. Let us take for example, the following record and perform a calculation upon it.

The record could be as shown in Figure 1

A short Basic program, normally a subroutine, such as that in Figure 2 could be used to work out the *HASH(No.)*

**NOTE** although I have used variable names of more than two characters length, Commodore Basic will only recognise the first two letters.

The variable MF (multiplication factor) is calculated to give the appropriate range of *HASH(No.)* numbers and the SF (subtraction factor) lowers the range minimum to zero. The range must generally be twice the total number of possible records to be written - this reduces the chance of a double up of the unique *HASH(No.)* numbers.

Lets us take another working example. If we apply the formula in the program in Figure 2 we will find that the name HENRY BLOGGS produces a *HASH(No.)* of 791. The name JULIA FORSE produces a *HASH(No.)* of 1359.

## Writing data

Right, how do we use this *HASH(No.)*? We find the next available record number

(from a sequential file named LASTUSED.DAT) and write the six fields of data into that record number in the main datafile, MAINDATA.DAT. We then calculate *HASH(No.)* and write the used record number as DATA into record no *HASH(No.)* in the index file INDEX.DAT.

Well, that may seem complicated but by the careful use of files a very retrievable database can be structured.

## Getting it back

Reading records is a reverse of the above although a little simpler. the user is asked for the SURNAM\$ and FIRNAM\$ of the record that is required to be retrieved. The *HASH(No.)* is then calculated with exactly the same formula, it will be the same as it was when the record was written as nothing has changed in the calculations. The data is then read from record number *HASH(No.)* in INDEX.DAT. That data will be the record number that the six fields of DATA were written to in MAINDATA.DAT so it only remains to read that data from MAINDATA.DAT.

If you can foresee that you may need to search for data by other fields or combinations of fields, more index files

**Figure 1**

FIELDNO	FIELDNAME	FIELDTYPE	DATA
1	SURNAM\$	ALPHA	BLOGGS
2	FIRNAM\$	ALPHA	HENRY
3	BIRDAT\$	ALPHA	200953
4	STREET\$	ALPHA	56 THE CLOSE
5	TOWN\$	ALPHA	HORNCHURCH
6	COUNTY\$	ALPHA	ESSEX



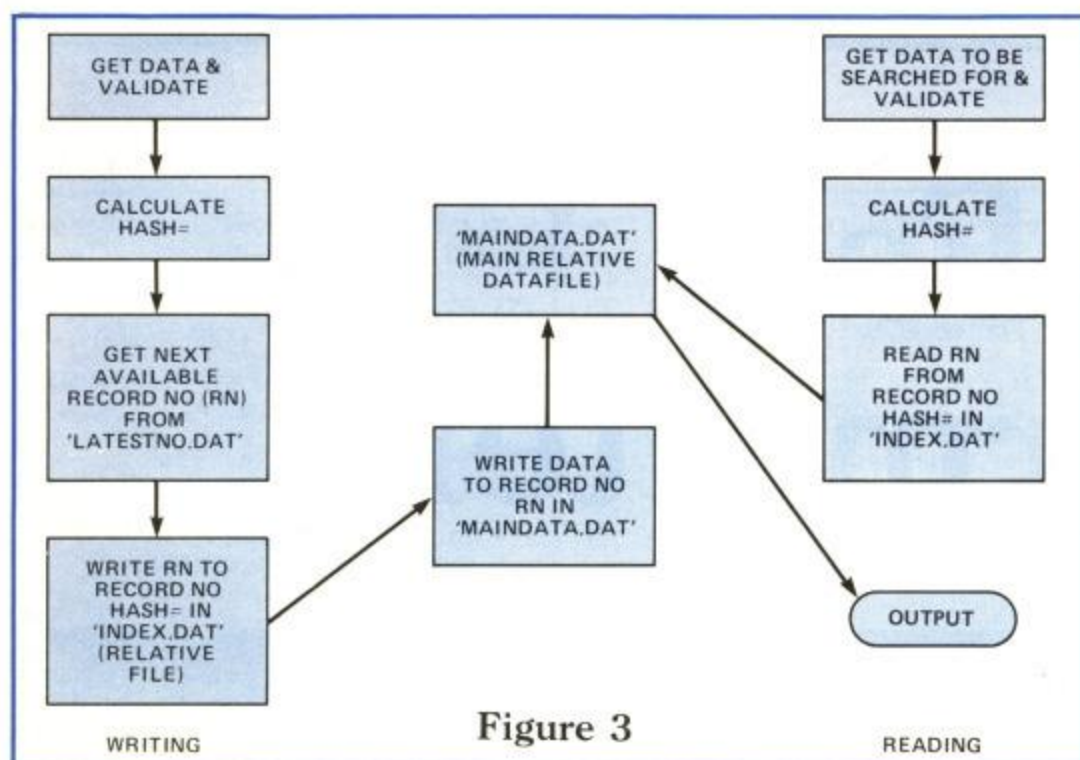


Figure 3

could be maintained – these could even be created at a later date by reading the MAINDATA.DAT file sequentially and creating additional index hash files.

## Duplicates

Now, one obvious problem is that of a *HASH(No.)* occurring more than once. To safeguard against this possibility, before writing to record number *HASH(No.)* in INDEX.DAT, read that record to ensure that no data exists. If it does, go down and read the next record and when you find a blank one, use that. Accordingly this means that after looking up a record in INDEX.DAT and reading the pointed record from MAINDATA.DAT, you must compare it with the specified data to ensure that it is the correct one. If it is not, go back to INDEX.DAT and read the next *HASH(No.)* down and read the record pointed to in MAINDATA.DAT. Keep doing this until you find the match.

Mathematics will show you that the chances of a double up of a *HASH(No.)* are unlikely as long as you have a maximum *HASH(No.)* of two times the maximum number of records to be written into MAINDATA.DAT.

## Getting MF and SF

To calculate MF and SF, one must define the characters that we are going to hash and index against. Take the example given. The maximum *HASH(No.)* will occur if the name is ZZZZZZ ZZZZZZ (SURNAM\$ and FIRNAM\$ respectively). The minimum *HASH(No.)* will occur if the name is AAAAAA

AAAAAA (SURNAM\$ and FIRNAM\$ respectively).

To calculate the range of possible *HASH(No.)*, one must decide the maximum number of records to be written to MAINDATA.DAT. Let's take

## Figure 2

```

10  SURNAM$="BLOGGS": FIRNAM$="HENRY"
20  HASHSTRING$=LEFT$(SURNAM$,3)+LEFT$(FIRNAM$,3)
30  PRINT HASHSTRING$: REM IT SHOULD BE "BLOHEN"
40  FORI=1TO6:H(I)=ASC(MID$(HASHSTRING$,I,I+1)):NEXTI
50  MF=6.641E-11: SF=724: REM SEE TEXT
60  HASH(No.)=(INT((H(1)*H(2)*H(3))+(H(4)*H(5)*H(6))*MF))-SF:
    REM DO THE CALCULATION
70  PRINT HASH(No.): REM IT SHOULD BE 791

```

the maximum here as 1200 records. We therefore require a *HASH(No.)* range of 0000–2400 and as each record in INDEX.DAT will occupy four bytes (2400 has four characters), the total space to be occupied by INDEX.DAT will be  $2400 \times 4 = 9600$  bytes.

The variable SF is chosen to move the range of minimum *HASH(No.)* – maximum *HASH(No.)* to 0000–2400.

O.K. this may all seem rather complex and time consuming but if you have any doubts to the speed of this system try the following:

## VARIABLES:

H(1) Is the left most character from the string SURNAM\$  
H(2) Is the second character from the string SURNAM\$  
H(3) Is the third character from the string SURNAM\$  
H(4) Is the left most character from the string FIRNAM\$  
H(5) Is the second character from the string FIRNAM\$  
H(6) Is the third character from the string FIRNAM\$

Set up a dummy data file with six fields of random dummy data in each record (have 1000 records). Hide a record of known data near the end of it at say, record number 941. Now, OPEN and READ the file in sequential order from record number one and at each record compare the read data with that known to be 'hiding'.

Go away and have lunch and if you're lucky the experiment may have found the 'hidden' record by the time you return.

Try setting up a small system as described above with an indexed HASH file and RUN it. You will now appreciate what relative data files and good indexing is all about!

## Multiple hash files

A word of caution. Before deciding to index every field or every combination of fields consider what the most likely unique fields will be. Index files occupy a significant amount of disk space and I

found it unnecessary in the example database to index more than two combinations. I have indexed SURNAM\$/FIRNAM\$ together and SURNAM\$/BIRDAT\$ together as it is very unlikely to find two BLOGG's with the same BIRDAT\$. In fact on a base of 27000 records, no two have ever matched in that way. So, despite Commodore's deplorably slow DOS, quite a workable database has been created.

The simple flowchart in Figure 3 summarises the use of hash files, both writing or creating and reading.



# Fast Formatter

*Fed up with waiting for your disks to be formatted? Speed things up with this handy program.*

**F**ast Formatter is a utility that, quite simply, fully formats a disk with ID in a fraction under 10 seconds.

The routine will no doubt gain most benefit when included within a database/filing program that calls upon a disk routine before saving file data. Provision for listing the directory of a disk without destroying a program in memory is also made.

## Getting it in

The program is presented here in the form of a Basic loader. This should be typed in using the *SYNTAX CHECKER* program that can be found on the *LISTINGS* page of this magazine. Once you RUN the program it will POKE the necessary machine code into memory. Should you want to SAVE the machine code for later retrieval by another program etc. then you can do so with the following instructions:

```
POKE43,0:POKE44,202:POKE45,168:
POKE46,207:SAVE"ML".8
```

## Using the program

The routine is screen-option driven and should cause no problems. To activate the routine a SYS call is required. To start

the program simply type:

SYS 51712

Now you can format a blank disk without the normal 80-second wait!!

PROGRAM: FAST FORMATTER	
30	4 REM * 10 SECOND FORMATTER
25	8 REM * RESIDENT IN HIGH ME
BF	12 REM * TO ACTIVATE ROUTINE
F2	16 REM * 'SYS 51712'
EE	20 REM * WILL RESET TO WARM
C3	24 REM * START WHEN FINISHED
0E	28 REM * THUS KEEPING BASIC
93	32 REM * MEMORY INTACT.
C9	36 REM * BY BEN WELLBAY 1986
DA	40 REM * FOR 'YOUR COMMODORE
01	44 FOR I=51712 TO 53160: READ A: P
F1	48 IF C<>176818 THEN PRINT "ERRO
	R IN DATA!":END
18	50 DATA 76,126,205,165,186,2
	01,8,176,4,169,8,133,186,169
	,5,162
73	51 DATA 207,160,202,32,134,2
	02,169,6,162,183,160,203,32,
	134,202,169
E2	52 DATA 7,162,169,160,204,32
	,134,202,169,1,162,192,160,0
	,141,202
79	53 DATA 202,142,200,202,140,
	201,202,169,87,141,199,202,3
	2,123,202,160
93	54 DATA 0,185,197,202,32,168
	,255,200,192,6,208,245,160,0
	,173,167
7B	55 DATA 2,32,168,255,32,174,
	255,162,0,160,5,142,192,202,
	140,193
60	56 DATA 202,32,123,202,160,0
	,185,189,202,32,168,255,200,
	192,5,208
98	57 DATA 245,32,174,255,234,3
	2,123,202,76,174,255,165,186
	,32,177,255
93	58 DATA 169,111,32,147,255,9
	6,141,185,202,134,251,132,25
	2,160,0,140
D4	59 DATA 184,202,32,123,202,1
	62,0,189,181,202,32,168,255,
	232,224,6

By Ben Wellbay



AF	60 DATA 208,245,162,32,177,2 51,32,168,255,200,202,208,24 7,32,174,255	0F	87 DATA 3,141,1,28,200,202,2 08,243,132,50,162,8,80,254,1 84,169	3A	113 DATA 76,126,205,136,16,2 48,76,126,205,169,147,32,22, 231,96,234
E3	61 DATA 192,0,208,219,96,77, 45,87,224,7,32,234,234,77,45 ,69	2F	88 DATA 85,141,1,28,202,208, 245,169,255,162,5,80,254,184 ,141,1	75	114 DATA 234,234,234,234,234 ,234,234,234,234,234,234 ,234,234,234
F3	62 DATA 0,5,234,234,234,77,4 5,87,192,0,1,234,234,234,234 ,173	43	89 DATA 28,202,208,247,162,1 87,80,254,184,189,0,1,141,1, 28,232	0A	115 DATA 234,234,234,169,1,1 33,212,160,0,32,207,255,153, 144,207,200
88	63 DATA 121,5,133,18,173,122 ,5,133,19,169,1,133,34,169,1 0,133	D7	90 DATA 208,244,160,0,80,254 ,184,177,48,141,1,28,200,208 ,245,169	02	116 DATA 201,13,208,245,192, 18,144,5,32,114,206,240,230, 136,152,240
AS	64 DATA 186,120,173,0,28,9,4 ,141,0,28,169,45,133,74,32,1 47	97	91 DATA 85,162,8,80,254,184, 141,1,28,202,208,247,198,188 ,208,154	C5	117 DATA 232,200,169,160,153 ,143,207,192,18,144,246,136, 185,144,207,153
00	65 DATA 5,198,74,208,249,162 ,0,32,160,5,32,189,5,169,238 ,141	C2	92 DATA 80,254,184,80,254,18 4,76,0,7,32,0,254,165,192,20 8,3	28	118 DATA 54,203,136,16,247,9 6,169,1,133,212,160,0,32,207 ,255,153
E2	66 DATA 12,28,32,0,6,133,192 ,173,0,28,41,251,141,0,28,16 9	89	93 DATA 76,103,7,169,200,133 ,189,165,67,133,188,169,0,13 3,50,32	84	119 DATA 144,207,200,201,13, 208,245,192,3,240,6,32,110,2 06,76,70
92	67 DATA 236,141,12,28,88,144 ,1,96,32,148,7,169,18,133,6, 169	68	94 DATA 119,7,162,10,164,50, 80,254,184,173,1,28,217,0,3, 208	3F	120 DATA 206,173,144,207,141 ,72,203,173,145,207,141,73,2 03,96,192,1
0B	68 DATA 0,133,7,32,200,7,32, 190,7,169,255,141,1,3,230,7	9D	95 DATA 48,200,230,50,202,20 8,239,32,119,7,160,187,80,25 4,184,173	90	121 DATA 240,251,169,20,32,2 2,231,136,208,248,96,169,0,1 33,212,169
F2	69 DATA 32,200,7,76,5,208,66 ,76,65,78,75,160,160,160,160 ,160	02	96 DATA 1,28,217,0,1,208,26, 200,208,242,162,252,80,254,1 84,173	74	122 DATA 147,32,210,255,234, 234,234,234,169,1,133,183,16 9,207,133,188
68	70 DATA 160,160,160,160,160, 160,160,160,66,87,160,50,65, 160,160,160	43	97 DATA 1,28,209,48,208,11,2 00,202,208,242,198,188,208,1 93,76,103	54	123 DATA 169,134,133,187,169 ,96,133,185,32,213,243,165,1 86,32,180,255
69	71 DATA 160,160,2,174,0,28,2 32,32,160,5,136,208,246,230, 34,76	EB	98 DATA 7,198,189,208,178,19 8,186,240,3,76,0,6,169,3,56, 96	88	124 DATA 165,185,32,150,255, 169,0,133,144,160,3,132,183, 32,165,255
15	72 DATA 189,5,160,2,174,0,28 ,202,32,160,5,136,208,246,96 ,138	11	99 DATA 165,34,201,35,240,6, 32,130,5,76,0,6,169,1,24,96	81	125 DATA 133,195,32,165,255, 133,196,164,144,208,53,164,1 83,136,208,235
A7	73 DATA 41,3,133,187,173,0,2 8,41,252,5,187,141,0,28,169, 4	A2	100 DATA 169,208,141,5,24,16 9,3,44,5,24,16,12,44,0,28,48	C5	126 DATA 160,6,169,32,32,22, 231,136,208,250,166,195,165, 196,32,205
27	74 DATA 133,187,162,0,202,20 8,253,198,187,208,249,96,165 ,34,32,75	79	101 DATA 246,173,1,28,184,16 0,0,96,104,104,76,88,7,32,19 0,7	6C	127 DATA 189,169,32,32,22,23 1,32,165,255,166,144,208,19, 201,0,240
66	75 DATA 242,138,10,10,10,10, 10,133,68,173,0,28,41,159,5, 68	3F	102 DATA 169,3,133,110,32,18 3,238,160,27,185,103,5,153,1 44,3,136	F2	128 DATA 6,32,22,231,76,214, 206,169,13,32,22,231,160,2,2 08,187
45	76 DATA 141,0,28,96,173,12,2 8,41,31,9,192,141,12,28,169, 255	6D	103 DATA 16,247,169,65,141,2 ,3,169,42,141,3,3,169,17,141 ,72	47	129 DATA 76,66,246,32,228,25 5,208,251,32,228,255,240,251 ,201,89,208
50	77 DATA 141,3,28,141,1,28,96 ,165,34,32,75,242,133,67,32, 213	F7	104 DATA 3,169,252,141,73,3, 96,160,0,152,153,0,3,200,208 ,250	3A	130 DATA 1,96,201,78,208,237 ,169,1,96,234,234,234,234,23 4,234,234
81	78 DATA 5,169,255,141,1,28,1 69,0,133,188,170,168,165,57, 153,0	4A	105 DATA 96,169,144,133,0,16 5,0,48,252,96,234,234,234,23 4,160,0	3C	131 DATA 234,234,234,234,234 ,234,234,234,234,234,234,234 ,234,234,96,234
6A	79 DATA 3,165,188,153,2,3,16 5,34,153,3,3,165,19,153,4,3	57	106 DATA 185,32,207,240,6,32 ,22,231,200,208,245,168,185, 59,207,240	13	132 DATA 147,13,32,70,79,82, 77,65,84,84,69,82,32,79,80,6 9
03	80 DATA 165,18,153,5,3,169,1 5,153,6,3,153,7,3,169,0,89	78	107 DATA 6,32,22,231,200,208 ,245,32,19,206,160,0,185,74, 207,240	84	133 DATA 82,65,84,73,79,78,6 5,76,32,58,0,13,32,68,73,83
48	81 DATA 2,3,89,3,3,89,4,3,89 ,5,3,153,1,3,24,152	9A	108 DATA 6,32,22,231,200,208 ,245,32,70,206,32,9,207,160, 1,132	17	134 DATA 75,32,78,65,77,69,3 2,42,32,0,13,32,68,73,83,75
7A	82 DATA 105,8,168,230,188,16 5,188,197,67,144,193,152,72, 232,138,157	3D	109 DATA 212,185,88,207,240, 6,153,240,4,200,208,245,32,3 ,202,160	D3	135 DATA 32,73,46,68,46,42,3 2,0,32,6,15,18,13,1,20,20
26	83 DATA 0,4,232,208,250,169, 75,141,0,4,169,3,133,49,32,4 8	CB	110 DATA 0,185,120,207,240,6 ,153,240,4,200,208,245,32,24 3,206,208	F9	136 DATA 9,14,7,46,46,32,0,3 2,1,14,15,20,8,5,18,32
7F	84 DATA 254,104,168,136,32,2 29,253,32,245,253,169,4,133, 49,32,233	FF	111 DATA 3,32,123,206,160,0, 185,103,207,240,6,153,240,4, 200,208	8C	137 DATA 4,9,19,11,32,63,32, 0,32,3,1,20,1,12,15,7
4F	85 DATA 245,133,58,32,143,24 7,169,0,133,50,169,255,141,1 ,28,162	CA	112 DATA 245,32,243,206,208, 19,32,9,207,234,234,234,160, 0,132,212	38	138 DATA 21,5,32,63,32,0,36, 14,254,246,83,0,0,0,0,0
41	86 DATA 5,80,254,184,202,208 ,250,162,10,164,50,80,254,18 4,185,0			90	139 DATA 66,87,13,160,160,16 0,160,160,160,160,160,160,16 0,160,160,160
				E2	140 DATA 160,160,0,255,0,2,0 ,16,255



# MULTIFILE 64

*A database that can easily be tailored to your own needs.  
For C64 plus disk drive.*

**M**ultifile is a disk-based data filing program. It has been constructed to offer the routines necessary for such a program, such as input and retrieval of data, as well as printing out neatly in columns. However it has been designed to allow it to be easily tailored to suit your own requirements. It could easily be converted to handle names and addresses (for personal use or club records), a collection catalogue, a stock control index, in fact the possibilities are endless.

Storing large amounts of array data in Basic introduces large time delays in array handling, so the main data storage routines of this program have been written in machine code. However, the main program is in Basic for ease of customisation for your own use.

*Multifile* comprises of two programs; the first is a Basic loader program for the machine code section, this loads the machine code (in a series of DATA statements) into memory at 49152. When the data is correct, the machine code section (just under 1k) is SAVED as a machine code file to disk. The main program can then load this file directly, and the loader with the machine code data need only be used once, hence saving time when the program is in use.

The second program is the Basic section of the filing system itself, and provides easy access to the machine code routines used.

## Data storage

The data is stored by the program in a series of records, each record comprising of a series of data items about one particular item. Each of these data items is called a field. The data is stored as in Figure 1.

*Multifile* will handle up to 10 fields, and up to 255 records in a file. Field length is not fixed, but the total length of all fields must not exceed 255 characters (this should not be a problem if a printout of the file on a standard 80-column printer is desired). The number of records in the file is updated as the program is used, but the number of fields required and their lengths must be set up as part of the program before any data files are created. Several versions of the program would be required to be kept if the program was used for a variety of filing applications.

## Setting up for use

The only lines needing to be changed in the Basic program are lines 500-560 (though the program title in lines 210 and 1000 could be altered also), and these should be changed to your requirements as you enter the listing. Line 500 defines variable F, the number of fields in the files handled. Lines 510-520 define the length (ie. number of characters) of each of the fields (if F is less than 10, the unused field lengths should be set to zero), and lines 530-560 define the titles of each of the fields. These titles are used within the program to refer to the columns, and are also printed as a header on any printout of the file. Note that the length of the title must be the same as the corresponding field size, otherwise an error will occur on running the program. If necessary, the field titles should be filled out with spaces eg. if FL(2)= 10, then FT\$(2)="SURNAME...". With these variables defined correctly, the program is ready to run.

The memory used by the program is as follows: 6 bytes + F bytes (F=number

of fields) + record storage. Each record uses memory as follows: 2 bytes + F bytes + 1 byte for each character in each field. All records are stored as string (ie. character) data, including any numbers stored. Memory is hence used more efficiently (and more quickly) than from Basic.

Data is stored upward from location 20000 (\$4E20), giving a maximum storage capacity of just over 20k per file. Note that records are not necessarily stored in memory in numerical order, though this is transparent to the user ie. they always appear to be in order when listed. Data is saved as a block of program memory from 20000 to the end of the file.

## Using the program

Upon RUNning the program, the machine code section is loaded in from disk if it is not already in memory. A menu of functions is then presented, and offers the following options (note that if entry to options is made in error, pressing RETURN from most prompts causes a return to the previous menu) :

### View Data in File

After choosing the output device (screen or printer (device 4)), or exit back to the menu, all of the data file currently in memory is listed out. The field titles are printed at the top, with the fields neatly printed out in vertical columns beneath their headings with one space between each column. The number of each record is printed down the left side. Listing can be slowed by pressing the CTRL key, and can be paused completely by holding SHIFT or clicking SHIFT LOCK. Pause is indicated by a red border, and the listing will continue when SHIFT is released.



## HOW IT WORKS.

10-410	Titles and load machine code.
500-590	Set up data lengths and check.
600-620	Set up file memory area.
700-790	Set up addresses of machine code routines.
1000-1120	Print main menu and get selection.
1500-1570	Print out data to screen or printer.
2000-2140	Add record to file.
2500-2590	Change record in file.
3000-3080	Delete record from file.
3500-3990	Disk file handling procedures.
4000-4390	Data processing routines.
4500-4540	Exit program.

### Subroutines

6000-6080	Input new or modified record.
7000-7070	Get disk filename and store in memory.
8000-8030	Read record from machine code buffer.
8500-8620	Check for empty file memory.
9000-9050	Place record into machine code buffer.
9500-9700	Wait for SPACE to be pressed.
9800-9850	Check disk drive error channel and report.

### Add a Record to File.

If more than one record is in the file already, the program asks for the number of the record after which the current one is to be added. Each of the field titles is then given, and an input is requested for this field. The whole record is then stored in memory by the machine code routine.

### Change Record in File.

The program prompts for the number of the record to be changed, and withdraws this record from memory. Each field title is then given, along with the current entry in this field, and a new entry is prompted for. If no change is required to this entry,

then pressing RETURN on it's own will indicate this. When all fields have been done, the old record is deleted from memory, and the new one added.

### Delete Record from File.

The program prompts for the number of the record to be deleted, and it is deleted from memory and printed on the screen.

### Disk File Handling.

Selecting this option presents another menu, offering file handling options as follows:

#### Load Data File

A file name is requested (the .FILE

extension should not be given), then the required file is loaded from disk. If a file transfer error occurs, this is indicated, otherwise a successful LOAD is indicated. After LOADING, a check is made to see if the file is compatible with the field lengths set up within the program, and if non-compatibility is found, a warning is given on the screen.

#### Save Data File

A file name is requested, then the data file currently in memory is SAVED to disk. If a file transfer occurs, this is indicated, otherwise a successful SAVE is indicated. Note that the extension .FILE is added to the given file name to indicate in the disk directory that this is a data file.

#### Disk Directory

A directory of the current disk is displayed on the screen. Pressing CTRL will slow the listing, and pressing SHIFT will pause the listing.

#### Rename a File

The file's current name is requested, followed by it's new one, the file is then renamed on the disk.

#### Delete a File

The file's name is requested, and the file deleted from the disk.

#### Return to Main Menu

The program returns immediately to the main menu screen.

#### Process Data.

Selecting this option presents another menu, offering data processing options as follows:

#### Search for Data

A search string of between 2 and 40 characters is asked for, the memory is scanned for all occurrences of this string. All records containing the search string are printed out in full along with their numbers, and a total number of finds is printed.

#### Sum column

A menu of field titles is printed, and a prompt for one of these is given. All elements in the requested column are then added up, and a total and average for the column are printed. This process may take some time for a long data file.

#### Return to Main Menu

The program returns immediately to the main menu screen.

#### Exit Program.

After asking for confirmation of exit, the program exits back to Basic.

Figure 1

		FIELDS			
	(No.)	NAME	ADDRESS	PHONE	MEMBER
R	001	J.Adams	3,Main St,	123456	654
E			Anytown.		
C	002	F.Jones	10,High St,	987654	321
O			Uptown.		
R	003	G.Smith	54,New St,	888666	432
D			Downtown.		
S	004	P.Young	17,Low Rd,	765765	223
			Anytown.		



## MULTIFILE MC GEN

```

41 10 REM *** MULTIFILE MACHINE
    CODE GENERATOR ***
AB 20 REM *** BY IAIN MURRAY (C
    ) 1986 ***
08 30 REM *** FOR YOUR COMMODORE
    E ***
34 50 POKE 53280,6:POKE 53281,1

1C 60 PRINT "[CLR, BLACK, DOWN3, R
    IGH3, RVSON] MULTIFILE MACHI
    NE CODE GENERATOR [RVSOFF]"
1B 70 PRINT "[DOWN3, RIGHT3] THIS
    WILL SAVE 'FILECODE' TO DIS
    K"
7F 80 PRINT "[DOWN5, RIGHT7] LOADI
    NG DATA - PLEASE WAIT"
E3 100 AD=49152:C=0
11 110 READ A:POKE AD,A:C=C+A:A
    D=AD+1
B4 120 IF AD<50135 THEN 110
BE 130 IF A=195 AND C=130189 TH
    EN 150
9A 140 PRINT "[DOWN3, RIGHT7] ERR
    OR IN DATA!!":END
41 150 PRINT "[DOWN3, RIGHT5] DAT
    A OK - PRESS [RVSON] SPACE [
    RVSOFF] TO SAVE"
8A 160 GET AS:IF AS<>" " THEN 1
    60
E9 170 POKE 43,0:POKE 44,192:PO
    KE 45,220:POKE 46,195
DB 180 SAVE "FILECODE",8
3E 190 END
64 1000 DATA 165,251,56,233,1,1
    33,251,165,252,233
7B 1010 DATA 0,133,252,96,165,2
    51,24,105,1,133
BE 1020 DATA 251,165,252,105,0,
    133,252,96,169,44
EC 1030 DATA 133,251,169,78,133
    ,252,96,160,0,177
51 1040 DATA 251,201,255,208,7,
    200,177,251,201,255
33 1050 DATA 240,6,32,14,192,76
    ,37,192,200,177
61 1060 DATA 251,96,173,2,206,2
    08,12,160,255,169
83 1070 DATA 1,141,3,206,169,3,
    76,83,192,160
B3 1080 DATA 7,169,4,170,32,186
    ,255,169,0,32
D9 1090 DATA 189,255,32,192,255
    ,169,3,24,109,2
38 1100 DATA 206,170,32,201,255
    ,76,101,193,32,231
74 1110 DATA 255,169,0,141,3,20
    6,96,169,3,141
1A 1120 DATA 5,206,24,109,32,78
    ,141,4,206,169
41 1130 DATA 0,141,6,206,172,4,
    206,177,251,32
CB 1140 DATA 210,255,238,4,206,
    238,6,206,172,5
7B 1150 DATA 206,177,251,205,6,
    206,208,232,172,5
CF 1160 DATA 206,185,31,78,56,2
    33,1,205,6,206
51 1170 DATA 48,20,152,56,233,2
    ,205,32,78,240
6A 1180 DATA 11,169,32,32,210,2
    55,238,6,206,76
92 1190 DATA 158,192,169,32,32,
    210,255,238,5,206
82 1200 DATA 173,5,206,56,233,3
    ,205,32,78,208
BA 1210 DATA 174,169,13,32,210,
    255,32,210,255,96
57 1220 DATA 255,255,32,62,192,
    240,1,96,169,1
CD 1230 DATA 141,7,206,32,94,19
    5,32,28,192,32
40 1240 DATA 37,192,201,255,240
    ,46,205,7,206,240
A3 1250 DATA 6,32,14,192,76,239
    ,192,32,41,195
B6 1260 DATA 32,60,195,32,117,1
    92,238,7,206,173
5D 1270 DATA 142,2,201,1,208,8,
    169,2,141,32
BE 1280 DATA 208,76,13,193,169,
    6,141,32,208,76
EB 1290 DATA 236,192,76,108,192
    ,255,0,169,8,170
93 1300 DATA 160,255,32,186,255
    ,173,0,207,162,1
51 1310 DATA 160,207,32,189,255
    ,32,28,192,32,37
92 1320 DATA 192,201,255,240,6,
    32,14,192,76,62
8F 1330 DATA 193,32,14,192,32,1
    4,192,32,14,192
53 1340 DATA 166,251,164,252,16
    9,31,133,251,169,78
F7 1350 DATA 133,252,169,251,32
    ,216,255,32,183,255
25 1360 DATA 41,128,141,8,206,9
    6,255,255,169,8
E4 1370 DATA 170,160,255,32,186
    ,255,173,0,207,162
A9 1380 DATA 1,160,207,32,189,2
    55,169,0,32,213
DA 1390 DATA 255,76,101,193,255
    ,255,32,28,192,32
8F 1400 DATA 37,192,201,255,240
    ,24,205,1,206,240
47 1410 DATA 2,16,6,32,14,192,7
    6,143,193,24
A7 1420 DATA 105,1,145,251,32,1
    4,192,76,143,193
FF 1430 DATA 160,0,32,14,192,32
    ,14,192,173,1
87 1440 DATA 206,24,105,1,145,2
    51,32,14,192,185
0A 1450 DATA 0,205,145,251,200,
    204,0,206,208,245
EB 1460 DATA 169,255,145,251,20
    0,145,251,200,145,251
ED 1470 DATA 238,33,78,96,255,2
    55,32,28,192,32
D0 1480 DATA 37,192,205,9,206,2
    40,6,32,14,192
2C 1490 DATA 76,223,193,169,0,1
    41,2,206,32,62
5A 1500 DATA 192,32,117,192,32,
    108,192,174,32,78
27 1510 DATA 169,3,168,24,109,3
    2,78,24,113,251
6B 1520 DATA 200,202,208,249,14
    1,11,206,169,0,141
B9 1530 DATA 10,206,172,11,206,
    177,251,160,0,145
E5 1540 DATA 251,201,255,240,9,
    140,10,206,32,14
06 1550 DATA 192,76,20,194,238,
    10,206,173,10,206
C2 1560 DATA 201,3,240,6,32,14,
    192,76,20,194
AF 1570 DATA 32,28,192,32,37,19
    2,201,255,240,27
8E 1580 DATA 56,237,9,206,176,6
    ,32,14,192,76
90 1590 DATA 61,194,160,2,177,2
    51,56,233,1,145
44 1600 DATA 251,32,14,192,76,6
    1,194,206,33,78
EF 1610 DATA 96,0,255,169,48,13
    3,252,169,2,133
79 1620 DATA 253,169,0,133,144,
    169,36,133,251,169
BD 1630 DATA 251,133,187,169,0,
    133,188,165,253,133
8E 1640 DATA 183,169,8,133,186,
    169,96,133,185,32
4A 1650 DATA 213,243,165,186,32
    ,180,255,165,185,32
3F 1660 DATA 150,255,164,144,20
    8,70,160,6,132,251
D0 1670 DATA 32,165,255,166,252
    ,133,252,164,144,208
F3 1680 DATA 55,164,251,136,208
    ,238,164,252,32,205
E6 1690 DATA 189,169,32,32,210,
    255,32,165,255,72
17 1700 DATA 173,142,2,201,1,24
    0,249,104,166,144
63 1710 DATA 208,24,170,240,6,3
    2,210,255,76,184
98 1720 DATA 194,169,13,32,210,
    255,165,197,201,63
DF 1730 DATA 240,4,160,4,208,18
    8,32,66,246,96
92 1740 DATA 255,255,32,28,192,
    173,2,207,24,105
F2 1750 DATA 2,141,2,207,32,37,
    192,205,15,206
A4 1760 DATA 240,6,32,14,192,76
    ,242,194,173,32
CB 1770 DATA 78,170,160,3,204,2
    ,207,208,3,141
BF 1780 DATA 3,207,24,113,251,2
    00,202,208,241,24
44 1790 DATA 105,3,141,0,206,16
    0,3,177,251,153
BF 1800 DATA 253,204,200,204,0,
    206,208,245,96,160
5E 1810 DATA 0,185,12,206,32,21
    0,255,200,192,3
C3 1820 DATA 208,245,169,32,32,
    210,255,96,238,14
47 1830 DATA 206,173,14,206,201
    ,58,208,23,169,48
02 1840 DATA 141,14,206,238,13,
    206,173,13,206,201
AD 1850 DATA 58,208,8,169,48,14
    1,13,206,238,12
20 1860 DATA 206,96,169,48,141,
    12,206,141,13,206
31 1870 DATA 141,14,206,238,14,
    206,96,0,255,32
88 1880 DATA 94,195,162,0,142,4
    ,207,232,142,7
CF 1890 DATA 206,32,28,192,32,3
    7,192,201,255,208
AA 1900 DATA 1,96,205,7,206,240
    ,6,32,14,192
4E 1910 DATA 76,126,195,32,14,1

```



```

92,160,0,32,14
D4 1920 DATA 192,177,251,217,61
    .3,240,13,201,255
84 1930 DATA 208,240,32,60,195,
    238,7,206,76,126
64 1940 DATA 195,200,204,60,3,2
    08,230,32,0,192
FD 1950 DATA 160,0,177,251,201,
    255,208,245,200,177
59 1960 DATA 251,201,255,208,23
    8,32,41,195,32,117
6A 1970 DATA 192,32,60,195,238,
    7,206,238,4,207
7E 1980 DATA 76,123,195

```

## PROGRAM: MULTIFILE

```

68 10 REM *** MULTIFILE-CONVERT
    IBLE FILE PROCESSOR ***
AB 20 REM *** BY IAIN MURRAY (C
    ) 1986 ***
08 30 REM *** FOR YOUR COMMODORE
    ***
17 40 REM *** DISK-BASED FILE'S
    TORAGE ***
93 80 UP$=CHR$(147)+CHR$(142):D
    N$=CHR$(147)+CHR$(14)+CHR$(8
    ):REM * CASES
21 100 PRINT "[BLACK]"+DN$:REM
    * BLACK/LOWER CASE
1E 110 POKE 53280,6:POKE 53281,
    1:REM * SCREEN COLOURS
AF 200 IF PEEK(49152)=165 THEN
    300
89 210 PRINT DN$+"[DOWN2,RIGHT1
    0,RVSON] [SM,SU,SL,ST,SI,SF
    ,SI,SL,SE] "
6A 220 PRINT "[DOWN4,RIGHT6,SL,
    SO,SA,SD,SI,SN,SG,SSPC,SM,SA
    ,SC,SH,SI,SN,SE,SSPC,SC,SO,S
    D,SE,SSPC,SS,SE,SC,ST,SI,SO,
    SN]"
D6 230 PRINT"[DOWN3,RIGHT13,RVS
    ON] [SP,SL,SE,SA,SS,SE] [SW,
    SA,SI,ST] [RVSOFF]"
6F 240 LOAD "FILECODE",8,1
1C 250 POKE 56,78:REM * MEMORY
    TOP
A0 300 DIM FL(10),FT$(10),FS$(1
    0)
18 400 MM = 20000:REM * START O
    F DATA AREA
9F 410 BF=52480:REM * M/C INTER
    FACE BUFFER
D7 500 F=***:REM * NUMBER OF FI
    ELDS
DE 510 FL(1)=***:FL(2)=***:FL(3
    )=***:FL(4)=***:FL(5)=***:RE
    M * LENGTH OF FIELDS
CB 520 FL(6)=***:FL(7)=***:FL(8
    )=***:FL(9)=***:FL(10)=***:R
    EM * LENGTH OF FIELDS
67 530 FT$(1)=[ST,SI,ST,SL,SE]
    1:FT$(2)=[ST,SI,ST,SL,SE]
    2:FT$(3)=[ST,SI,ST,SL,SE,
    SSPC]3:REM * FIELD TITLES
1F 540 FT$(4)=[ST,SI,ST,SL,SE]
    4:FT$(5)=[ST,SI,ST,SL,SE]

```

```

5:FT$(6)=[ST,SI,ST,SL,SE]
6:REM * FIELD TITLES
93 550 FT$(7)=[ST,SI,ST,SL,SE,
    SSPC]7:FT$(8)=[ST,SI,ST,SL
    ,SE,SSPC]8:FT$(9)=[ST,SI,S
    T,SL,SE,SSPC]9:REM * FIELD
    TITLES
B1 560 FT$(10)=[ST,SI,ST,SL,SE
    ,SSPC]10:REM * FIELD TITLES
EE 565 IF F<2 OR F>10 OR F<>INT
    (F) THEN PRINT "[SF]IELD NUM
    BER ERROR":END
9C 570 FOR I=1 TO F:IF LEN(FT$(
    I))<>FL(I) THEN PRINT "[SS]T
    RING":I:"LENGTH ERROR":END
86 580 FS$(I)=FT$(I):IF LEN(FS$
    (I))>12 THEN FS$(I)=LEFT$(FS
    $(I),12)
D2 590 NEXT
07 600 POKE MM,F:POKE MM+1,0:RE
    M * SET UP MEMORY AREA
F8 610 FOR I=1 TO F:POKE MM+1+I
    ,FL(I):NEXT
3C 620 FOR I=MM+12 TO MM+14:POK
    E I,255:NEXT
E3 700 PT=49374:REM *** PRINTOU
    T ***
F5 710 SV=49449:REM *** SAVE FI
    LE ***
BF 720 LD=49520:REM *** LOAD FI
    LE ***
0A 730 AR=49548:REM *** ADD REC
    ORD ***
80 740 DR=49628:REM *** DELETE
    RECORD ***
07 750 DY=49765:REM *** DISK DI
    RECTORY ***
C7 760 SR=49894:REM *** RECORD
    SEARCH ***
08 770 SH=50031:REM *** DATA SE
    ARCH ***
40 775 REM *** END OF CODE AT 5
    0138 ***
CB 780 FT$=[DOWN,RIGHT7,RVSON]
    FILE TRANSFER COMPLETED [RV
    SOFF]"
49 790 FE$=[DOWN,RIGHT9]FILE T
    RANSFER ERROR!"
8D 999 REM *** MAIN MENU ***
F4 1000 PRINT DN$+"[DOWN2,RIGHT
    13,RVSON] [SM,SU,SL,ST,SI,S
    F,SI,SL,SE] "
4E 1010 PRINT "[DOWN,RIGHT13,RV
    SON] [SM,SA,SI,SN] [SM,SE,S
    N,SU] "
32 1020 PRINT "[DOWN2,RIGHT8]1)
    [SSPC,SV]IEW [SD]ATA IN [SF]
    ILE"
9C 1030 PRINT "[DOWN,RIGHT8]2) [
    SSPC,SA]DD [SR]ECORD TO [SF]
    ILE"
BA 1040 PRINT "[DOWN,RIGHT8]3) [
    SSPC,SC]HANGE [SR]ECORD IN [
    SF]ILE"
59 1050 PRINT "[DOWN,RIGHT8]4)
    [SD]ELETE [SR]ECORD FROM [SF]
    ILE"
0C 1060 PRINT "[DOWN,RIGHT8]5)
    [SD]ISK [SF]ILE [SH]ANDLING"
9E 1070 PRINT "[DOWN,RIGHT8]6)
    [SP]ROCESS [SD]ATA"
DE 1080 PRINT "[DOWN,RIGHT8]7)
    [SE]XIT [SP]ROGRAM"
54 1100 PRINT "[DOWN2,RIGHT4,SW

```

```

]HICH DO YOU REQUIRE (1-7) ?
"
F2 1110 GET A$:A=VAL(A$):IF A<1
    OR A>8 THEN 1110
A4 1120 ON A GOTO 1500,2000,250
    0,3000,3500,4000,4500
6A 1499 REM *** VIEW FILE DATA
    ***
54 1500 PRINT "[CLR,DOWN2,RIGHT
    11,RVSON,SV]IEW [SD]ATA IN [
    SF]ILE"
F2 1505 GOSUB 8500:IF MX=0 THEN
    1000
CE 1510 PRINT "[DOWN2,RIGHT4,SO
    ]UTPUT TO [SS]CREEN OR [SP]R
    INTER, OR"
7B 1520 PRINT "[RIGHT4]RETURN T
    O [SM]AIN [SM]ENU ([SS]/[SP]
    /[SM])[SSPC]?"
BA 1530 GET A$:IF A$="M" THEN 1
    000
53 1534 DV=0:IF A$="P" THEN DV=
    1:GOTO 1538
31 1536 IF A$<>"S" THEN 1530
4C 1538 POKE 52738,DV
37 1540 PRINT "[DOWN2,RIGHT6,SP
    ]RESS [RVSON,SS,SH,SI,SF,ST,
    RVSOFF] TO PAUSE LISTING(DOW
    N2)"
0D 1550 IF DV=1 THEN PRINT "[RI
    GHT15,RVSON] [SP,SR,SI,SN,ST
    ,SI,SN,SG] [RVSOFF,DOWN]":OP
    EN 1,4,7:GOTO 1555
F6 1553 OPEN 1,3
47 1555 PRINT#1,"# ";:FOR I=1
    TO F:PRINT#1," ";FT$(I):NEX
    T:PRINT#1:PRINT#1:CLOSE1
1A 1560 SYS PT:IF PEEK(52744)<>
    0 THEN PRINT "[UP2,RIGHT9,SP
    ,SR,SI,SN,ST,SE,SR,SSPC,SN,S
    O,ST,SSPC,SA,SV,SA,SI,SL,SA,
    SB,SL,SE]!"
32 1570 GOSUB 9600:GOTO 1000
0D 1999 REM *** ADD DATA TO FIL
    E ***
63 2000 PRINT "[CLR,DOWN2,RIGHT
    12,RVSON,SA]DD [SD]ATA TO [S
    F]ILE[DOWN2]"
31 2010 PRINT "[DOWN2,RIGHT4,SD
    ]O YOU WISH TO ADD A RECORD
    ([SY]/[SN]) ?"
27 2020 GET A$:IF A$="N" THEN 1
    000
19 2030 IF A$<>"Y" THEN 2020
8F 2040 MX=PEEK(MM+1):IF MX=0 T
    HEN MX=1
91 2045 IF MX=1 THEN A=1:GOTO 2
    080
72 2050 PRINT "[DOWN2,RIGHT4,SW
    ]HICH RECORD DO YOU WISH TO
    ADD"
B8 2060 PRINT "[RIGHT4]AFTER (
    1 -":MX:") :[SSPC]":
C5 2070 INPUT A$:A=VAL(A$):IF A
    <1 OR A>MX THEN 2050
BF 2080 POKE 52737,A:IF PEEK(MM
    +1)=0 THEN POKE 52737,0
1D 2120 RP=0:GOSUB 6000
F7 2130 SYS AR
13 2140 PRINT "[DOWN2,RIGHT13,S
    R]ECORD [SA]DDED":GOSUB 9600
    :GOTO 1000
CD 2499 REM *** CHANGE A RECORD
    ***
EB 2500 PRINT "[CLR,DOWN2,RIGHT
    13,RVSON,SC]HANGE [SR]ECORD"

```



```

59 2510 GOSUB 8500:IF MX=0 THEN
1000
AE 2520 PRINT "[DOWN2,RIGHT4,SW
]HICH RECORD DO YOU WISH TO
CHANGE"
E3 2530 PRINT "[RIGHT4]( 1 -";M
X;") ";
04 2540 AS="":INPUT AS:IF LEN(A
$)=0 THEN 1000
26 2550 A=VAL(AS):IF A<1 OR A>M
X THEN 2520
CA 2560 POKE 52751,A:SYS SR:GOS
UB 8000:RP=1:GOSUB 6000
AC 2565 PRINT "[WHITE]";:REM *
WHITE FOR INVISIBILITY
3E 2570 POKE 52745,A:SYS DR:REM
* REMOVE OLD RECORD
26 2580 POKE 52737,A-1:SYS AR:R
EM * ADD NEW RECORD
30 2590 PRINT "[BLACK,UP,RIGHT1
0,SR]ECORD [SC]HANGED":GOSUB
9600:GOTO 1000
04 2999 REM *** DELETE A RECORD
***
74 3000 PRINT "[CLR,DOWN2,RIGHT
13,RVSON,SD]ELETE [SR]ECORD"

1F 3010 GOSUB 8500:IF MX=0 THEN
1000
31 3020 PRINT "[DOWN2,RIGHT4,SW
]HICH RECORD DO YOU WISH TO
DELETE"
59 3030 PRINT "[RIGHT4]( 1 -";M
X;") ";
AA 3040 AS="":INPUT AS:IF LEN(A
$)=0 THEN 1000
06 3050 A=VAL(AS):IF A<1 OR A>M
X THEN 3020
1D 3060 PRINT "[DOWN2]";POKE 52
745,A:SYS DR
B5 3070 PRINT "[DOWN,RIGHT12,SR
]ECORD [SD]ELETED"
B6 3080 GOSUB 9600:GOTO 1000
65 3499 REM *** DISK FILE HANDL
ING ***
5F 3500 PRINT DN$+"[DOWN2,RIGHT
11,RVSON,SD]ISK [SF]ILE [SH]
ANDLING[DOWN2]"
18 3510 PRINT "[RIGHT8]1) [SL]O
AD [SD]ATA [SF]ILE"
55 3520 PRINT "[DOWN,RIGHT8]2)
[SS]AVE [SD]ATA [SF]ILE"
F3 3530 PRINT "[DOWN,RIGHT8]3)
[SD]ISK [SD]IRECTORY"
6F 3540 PRINT "[DOWN,RIGHT8]4)
[SR]ENAME A [SF]ILE"
DF 3550 PRINT "[DOWN,RIGHT8]5)
[SD]ELETE A [SF]ILE"
F3 3560 PRINT "[DOWN,RIGHT8]6)
[SR]ETURN TO [SM]AIN [SM]ENU"

E2 3570 PRINT "[DOWN2,RIGHT4,SW
]HICH DO YOU REQUIRE (1-6) ?"

3E 3580 GET AS:IF AS<"1" OR AS>
"6" THEN 3580
8B 3590 A=VAL(AS):ON A GOTO 370
0,3800,3600,3900,3950,1000
5B 3600 PRINT UP$+"[DOWN2,RIGHT
4]PRESS [RVSON]SHIFT[RVSOFF]
TO PAUSE DIRECTORY[DOWN2]"
DB 3610 SYS DY:GOSUB 9500:GOTO
3500
AB 3700 PRINT UP$+"[DOWN2,RIGHT
15,RVSON]LOAD DATA"
AB 3710 GOSUB 7000:IF FM=0 THEN
3500
56 3740 GOSUB 9700:PRINT "[DOWN
,RIGHT15,RVSON] LOADING [RVS
OFF]"
4A 3750 SYS LD
40 3760 IF PEEK(52744)<>0 THEN
PRINT FE$:GOTO 3780
DB 3770 GOSUB 9800:IF ER<20 THE
N PRINT FT$
41 3775 IF ER>=20 THEN 3780
F9 3776 F=PEEK(MM+I+1)
62 3777 IF FL(1)<>LEN(FT$(I)) T
HEN PRINT "[DOWN,RIGHT10]INC
OMPATIBLE FILE!":GOTO 3780
52 3778 NEXT
68 3780 GOSUB 9500:GOTO 3500
3E 3800 PRINT UP$+"[DOWN2,RIGHT
15,RVSON]SAVE DATA"
2C 3810 GOSUB 8600:IF MX=0 THEN
3500
C1 3820 GOSUB 7000:IF FM=0 THEN
3500
27 3840 GOSUB 9700:PRINT "[DOWN
,RIGHT15,RVSON] SAVING [RVSO
FF]"
4A 3850 SYS SV
36 3860 GOTO 3760
BF 3900 PRINT UP$+"[DOWN2,RIGHT
14,RVSON]RENAME FILE"
A7 3910 PRINT "[DOWN2,RIGHT4]GI
VE PRESENT FILENAME:"
B1 3920 PRINT "[DOWN,SPC6]12345
67890123456"
FE 3925 INPUT "[SPC4]";CN$:FM$=
CN$:IF LEN(CN$)=0 THEN 3500
88 3927 IF LEN(CN$)>16 THEN 391
0
E2 3930 PRINT "[DOWN2,RIGHT4]GI
VE NEW FILENAME:"
22 3935 PRINT "[DOWN,SPC6]12345
67890123456"
A7 3940 INPUT "[SPC4]";NN$:IF L
EN(NN$)=0 THEN 3500
00 3942 IF LEN(NN$)>16 THEN 393
0
C6 3945 PRINT "[DOWN,RIGHT15,RV
SON] RENAMING [RVSOFF]"
A3 3949 OPEN 1,8,15,"R:"+NN$+"=
"+CN$:CLOSE 1:GOSUB 9800:GOS
UB 9500:GOTO 3500
FA 3950 PRINT UP$+"[DOWN2,RIGHT
14,RVSON]DELETE FILE"
02 3960 PRINT "[DOWN2,RIGHT4]GI
VE FILENAME:"
4F 3970 PRINT "[DOWN,SPC6]12345
67890123456"
4D 3980 INPUT "[SPC4]";CN$:FM$=
CN$:IF LEN(CN$)=0 THEN 3500
2A 3982 IF LEN(CN$)>16 THEN 396
0
30 3985 PRINT "[DOWN,RIGHT14,RV
SON] DELETING [RVSOFF]"
4A 3990 OPEN 1,8,15,"S:"+CN$:CL
OSE 1:GOSUB 9800:GOSUB 9500:
GOTO 3500
84 3999 REM *** PROCESS DATA **
*
B6 4000 PRINT "[CLR,DOWN2,RIGHT
14,RVSON,SP]ROCESS [SD]ATA"
65 4004 GOSUB 8500:IF MX=0 THEN
1000
CB 4010 PRINT "[DOWN2,RIGHT4]1)
[SS]EARCH FOR [SD]ATA"
A5 4020 PRINT "[DOWN,RIGHT4]2)
[SS]UM [SD]ATA [SC]OLUMN"

9A 4030 PRINT "[DOWN,RIGHT4]3)
[SR]ETURN TO [SM]AIN [SM]ENU"
8A 4040 PRINT "[DOWN2,RIGHT4,SW
]HICH DO YOU REQUIRE (1-3) ?"
46 4050 GET AS:A=VAL(AS):IF A<1
OR A>3 THEN 4050
19 4060 PRINT AS:"[DOWN]":ON A
GOTO 4100,4300,1000
89 4100 PRINT "[DOWN,RIGHT4,SG]
IVE SEARCH STRING:"
C1 4110 INPUT SS:S=LEN(SS):IF S
<2 OR S>40 THEN 4100
61 4120 PRINT "[DOWN]";POKE 828
,S:FOR I=1 TO S:POKE 828+I,A
SC(MID$(S$,I,1)):NEXT
E7 4130 SYS SH
77 4140 PRINT "[DOWN,RIGHT4,ST]
OTAL OF":PEEK(52996):"OCCURR
ENCES FOUND"
7C 4150 GOSUB 9600:GOTO 1000
C8 4300 FOR I=1 TO F STEP 2:PRI
NT "[RIGHT3]";I:"[LEFT]";:F
S$(I);
C4 4305 IF I+1>F THEN PRINT:GOT
O 4310
B5 4307 PRINT TAB(20);I+1:"[LEF
T]";:FS$(I+1)
06 4310 NEXT:PRINT "[DOWN,RIGHT
4,SW]HICH COLUMN DO YOU WISH
TO SUM"
AA 4315 PRINT "[RIGHT4]( 1 -";F
;") ";
6F 4320 INPUT AS:A=VAL(AS):IF A
<1 OR A>F THEN PRINT "[DOWN]
":GOTO 4300
3C 4330 N=0:PRINT "[DOWN2,RIGHT
15,RVSON] [SS,SU,SM2,SI,SN,S
G] [RVSOFF]"
90 4340 FOR W=1 TO MX:POKE 5275
1,W:POKE 52994,A:SYS SR
47 4350 RL=PEEK(BF+A-1):RX=PEEK
(52995)+BF:N$=""
39 4360 FOR J=0 TO RL-1:N$=N$+C
HR$(PEEK(RX+J)):NEXT
1D 4370 N=N+VAL(N$):NEXT
2B 4380 PRINT "[DOWN2,RIGHT4,SC
]OLUMN";A;("FS$(A);") TOTA
L =";N
7B 4385 PRINT "[DOWN,RIGHT4]([S
T]OTAL OF";MX;"ITEMS)"
5C 4390 PRINT "[DOWN,RIGHT4,SC]
OLUMN AVERAGE =";N/MX:GOSUB
9600:GOTO 1000
19 4499 REM *** EXIT PROGRAM **
*
27 4500 PRINT "[CLR,DOWN2,RIGHT
14,RVSON,SE]XIT [SP]ROGRAM[D
OWN2]"
64 4510 PRINT "[DOWN2,RIGHT4,SD
]O YOU WISH TO EXIT ([SY]/[S
N]) ?"
DD 4520 GET AS:IF AS="N" THEN 1
000
EE 4530 IF AS<>"Y" THEN 4520
56 4540 PRINT UP$+CHR$(9)+"[DOW
N2]":END
DE 5999 REM *** GET NEW DATA RE
CORD ***
F2 6000 FOR I=1 TO F
38 6020 IF RP=1 THEN PRINT "[DO
WN,RIGHT4,SE]XISTING ";FS$(I
);:IS="":PRINT "[RIGHT4]";D
$(I)
72 6025 IF RP=0 THEN D$(I)=""

```



```

B2 6030 A$="":PRINT "[DOWN,RIGH
T4,SN]EW ";FS$(I);" (SI)TEM
:"
15 6035 INPUT "[RIGHT4]";A$
7E 6040 IF LEN(A$)=0 THEN 6055
06 6044 IF LEN(A$)<=FL(I) THEN
6050
4B 6046 PRINT "[DOWN,RIGHT4,SD]
ATA (SI)NPUT (ST)OO (SL)ONG!
! (SSPC) ((SM)UST BE"
A8 6047 PRINT "[RIGHT3]";FL(I);
"CHARACTERS LONG OR LESS";G
OTO 6020
57 6050 D$(I)=A$:DL(I)=LEN(A$)
E0 6055 IF LEN(D$(I))=0 THEN PR
INT "[DOWN,RIGHT4,SN]O (SD)A
TA (SI)NPUT!";GOTO 6020
28 6060 NEXT I
E9 6070 GOSUB 9000
59 6080 RETURN
74 6999 REM *** GET FILENAME AN
D STORE ***
00 7000 PRINT "[DOWN2,RIGHT22]1
2345678901"
AF 7010 PRINT "[RIGHT4]GIVE FIL
ENAME : (SSPC)";
88 7020 FM$="":INPUT FM$:FM=LEN
(FM$)
A1 7025 IF FM=0 THEN 7070
70 7030 IF FM<2 OR FM>11 THEN 7
000
73 7040 FM$=FM$+".FILE"
69 7050 POKE 52992,FM+5
7C 7060 FOR I=1 TO FM+5:POKE 52

```

```

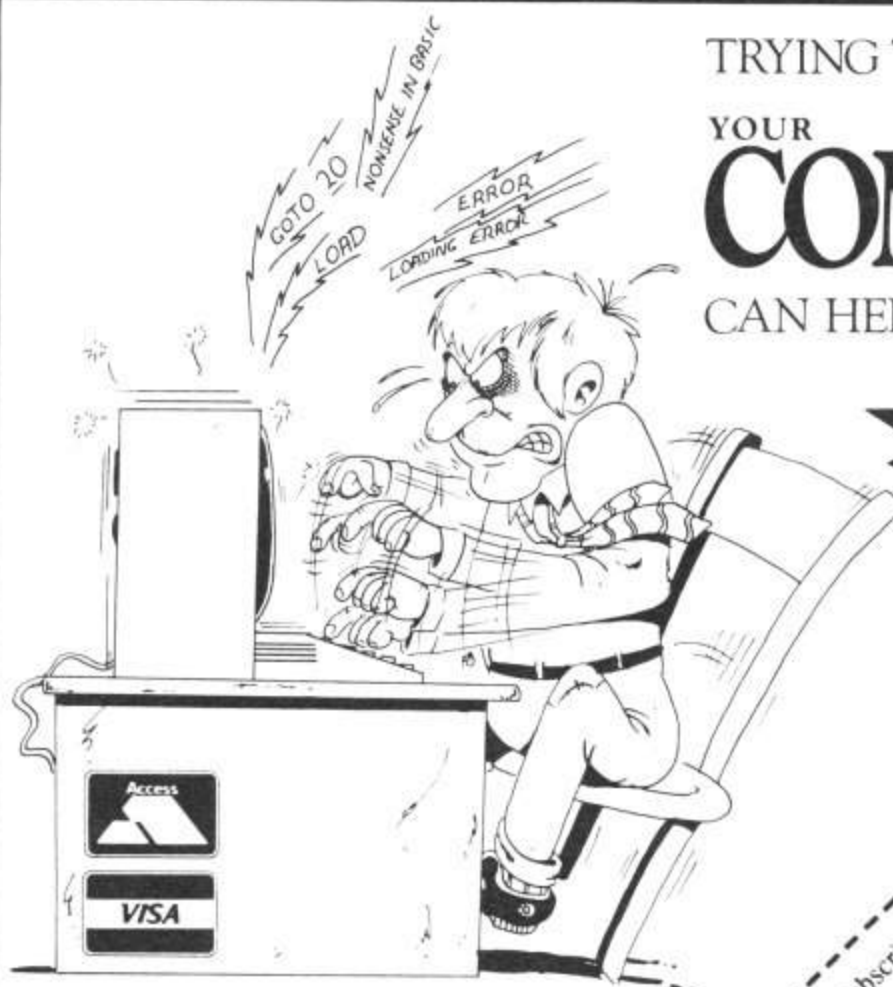
992+I,ASC(MID$(FM$,I,1)):NEX
T
37 7070 RETURN
75 7999 REM *** GET RECORD FROM
M/C BUFFER ***
9D 8000 PRINT "[DOWN,RIGHT13,RV
SON] (SP,SL,SE,SA,SS,SE,SSPC
,SW,SA,SI,ST) (RVSOFF)"
F7 8005 FOR I=1 TO F:DL(I)=PEEK
(BF+I-1):D$(I)="":NEXT
FC 8010 TT=0:FOR I=1 TO F
58 8020 FOR J=1 TO DL(I):D$(I)=
D$(I)+CHR$(PEEK(BF+F+TT+J-1)
):NEXT:TT=TT+DL(I):NEXT
F3 8030 RETURN
98 8500 MX=PEEK(MM+1)
2E 8510 IF MX=0 THEN PRINT "[DO
WN4,RIGHT10,SN,SO] (SR,SE,SC
,SO,SR,SD,SS) (SI,SN) (SF,SI
,SL,SE)";GOSUB 9600
E7 8520 RETURN
3C 8600 MX=PEEK(MM+1)
91 8610 IF MX=0 THEN PRINT "[DO
WN4,RIGHT10]NO RECORDS IN FI
LE!";GOSUB 9500
43 8620 RETURN
B4 8999 REM *** PUT RECORD INTO
M/C BUFFER ***
59 9000 PRINT "[DOWN,RIGHT13,RV
SON] (SP,SL,SE,SA,SS,SE,SSPC
,SW,SA,SI,ST) (RVSOFF)"
7A 9005 FOR I=1 TO F:POKE (BF+I
-1),DL(I):NEXT:TT=0:BL=F
BC 9010 FOR I=1 TO F
B6 9020 FOR J=1 TO DL(I):POKE (

```

```

BF+F+TT+J-1),ASC(MID$(D$(I),
J,1))
37 9030 NEXT:TT=TT+DL(I):BL=BL+
DL(I):NEXT
EE 9040 POKE 52736,BL:REM * BUF
FER LENGTH
F3 9050 RETURN
F6 9499 REM *** WAIT FOR SPACE
PRESS ***
61 9500 PRINT "[DOWN,RIGHT7]PRE
SS (RVSON) SPACE (RVSOFF) TO
CONTINUE"
39 9510 GET A$:IF A$<>" " THEN
9510
DB 9520 RETURN
2D 9600 PRINT "[DOWN,RIGHT7,SP]
RESS (RVSON) (SS,SP,SA,SC,SE
) (RVSOFF) TO (SC)ONTINUE"
B5 9610 GOTO 9510
2C 9700 PRINT "[DOWN,RIGHT11]PR
EPARE DISK AND":GOTO 9500
CF 9799 REM *** GET DISK STATUS
***
D3 9800 OPEN 15,8,15:INPUT#15,E
R,ER$:CLOSE 15
DE 9810 IF ER<20 THEN 9850
20 9815 PRINT "[UP2]";FE$
6E 9820 PRINT "[DOWN,RIGHT9]ERR
OR #";ER
11 9830 PRINT "[DOWN,RIGHT9](";
ER$;")"
50 9840 PRINT "[DOWN,RIGHT9]FIL
ENAME : ";FM$
2E 9850 RETURN

```



TRYING TO USE YOUR COMPUTER?...

YOUR  
**COMMODORE**  
CAN HELP.

£16.60  
for 12 Issues U.K.  
£21.50 for 12 Issues  
Overseas Surface Mail  
£57.00 for 12 Issues  
Overseas Air Mail

Send this form with your remittance to:  
**INFONET LTD., Times House, 179 The Marlowes,  
Hemel Hempstead, Herts. HP1 1BB.**

Please begin my subscription(s) to YOUR COMMODORE with the  
I enclose my cheque/postal order for £  
or debit £  
valid from  
NAME (Mr/Mrs/Miss)  
ADDRESS  
Postcode  
Signature  
Date  
made payable to Argus Specialist Publications Ltd.  
to  
issue.

Please use BLOCK CAPITALS  
and include post codes



# Disk File Descriptor

*Keep track of which files do what with this handy disk utility.*

**P**icture this scenario, you have just spent hours on your epic program, and have yet to tie the loose ends, but it's getting late, and you'll finish it off the next day. You SAVE your routine with all others, any old file name will do, and pack up for the night.

Unfortunately, the next day you're busy, and the day after you've just bought a cracker of a game and then spend four months getting an unbeatable score on it. You flick through your old disks 'cos you're running short on space, and you find that disk you worked on all that time ago. "I wonder what these files are, names like TRY1 and ML4, even a TZXE27!"

Your epic has gone, hasn't it! Has it? Not now! This little program will help you to remember even the most odd-named files you happened to SAVE with a description of up to 55 characters in length! A SYS for an old game perhaps? No need to remember it now, write it in the file descriptor and it will never get lost.

The Commodore DOS on the 64 is unfortunately limited in several ways, not least in the fact that file names may only have 16 characters. For most applications, it isn't enough, this program allows you to add a definitive description of 55 characters to any file on the disk. Merely create the descriptive titles and SAVE them onto the particular disk to which it refers. Then, simply LOAD and RUN Disk File Descriptor and, if you desire, additions and editions can be made at any time.

## Getting it in

Disk File Descriptor is a Basic program, and thus fairly simple to follow and type in. Use the SYNTAX CHECKER program that is found on the LISTINGS page to help you with your typing.

After RUNNING, the program loads the description file, if one exists, then LOADs the directory. On screen, you are presented with a menu offering five choices.

Option 1 views the directory together with the descriptions and a further menu at the bottom of the screen gives options to add or alter the current text. The file names are shown in blocks of six. If there are more files on disk keys f5 and f7 allow you to page back and forth as required. Pressing the f3 key puts you into edit mode.

Once into the edit mode you enter the file number (the number next to the files on the screen). The cursor then moves to the start of the light field in which you can make the necessary text edition. If you have made an incorrect choice of file then just press RETURN and the directory screen will return with no alterations made. All characters can be entered within the description area except inverted commas (quote marks) and although the cursor travels automatically to the next text line it cannot distinguish ends of words unlike a word processor. If you are a tidy sort of person, you will no doubt space your description so that it is easily readable at a later date. The INST/DEL key erases as per its normal function, and RETURN after completion of entering your desired text brings the directory screen back for you to continue entering descriptive text or, by pressing f1 and option 3, SAVE the description file onto disk. Make sure that enough room exists on the disk for this file.

As far as the other options are concerned, option 2 gives you normal DOS commands such as scratch, validate and rename. It also formats disks.

As stated, option 3 SAVES the directory descriptions to disk as a sequential file. If a file is already present on disk, the program will overwrite it with the cur-

rent data in memory. Choosing option 4 simply allows you to work with other disks and reRUNs the program as this option is chosen.

To exit the program, choose option 5.

I am sure you will find many uses for this program, and it would be interesting to hear how it could be adapted to individual needs.

### PROGRAM: DISK DESCRIPTOR

```

84 1 REM *****
*
27 2 REM * DISK FILE DESCRIPTOR
*
3B 3 REM * BEN WELLBAY
*
3B 4 REM * YOUR COMMODORE 1986
*
80 5 REM *****
*
CA 6 POKE53280,11:POKE53281,11:
PRINT"[CLR,DOWN6,RIGHT4,C8]D
ISK FILE DESCRIPTOR"
2C 7 PRINT"[DOWN2,RIGHT4]PLEASE
WAIT - READING DATA"
39 8 FORI=830TO902:READA:POKEI,
A:NEXT
90 9 A1$="[HOME,DOWN21]":FL=1
61 10 A2$="[RVSON,SPC19,RVSOFF,
SPC4,RVSON,SPC36]"
71 11 FA=144:DIMDT$(FA),J$(FA),
K$(FA),J1$(FA)
CE 12 FORI=0TOFA:DT$(I)=CHR$(32
)+"[SPC18]":NEXT
FE 13 PRINT"[UP,SPC11]READING D
IRECTORY ":OPEN1,8,0,"$":SY
S830:CLOSE1
83 14 GOSUB88
6F 15 POKE198,0:PRINT"[CLR,DOWN
]"SPC(6)"[RVSON][1][RVSOFF]"
SPC(3)"GET DISK DIRECTORY"
AB 16 PRINT"[DOWN2]"SPC(6)"[RVS
ON][2][RVSOFF]"SPC(3)"DISK D
RIVE COMMANDS"
DB 17 PRINT"[DOWN2]"SPC(6)"[RVS
ON][3][RVSOFF]"SPC(3)"SAVE D
ESCRIPTIONS FILE"
45 18 PRINT"[DOWN2]"SPC(6)"[RVS
ON][4][RVSOFF]"SPC(3)"CHANGE
TO ANOTHER DISK"
4B 19 PRINT"[DOWN2]"SPC(6)"[RVS

```

Ben Wellbay.



```

ON) [5] [RVSOFF] "SPC(3) "QUIT P
ROGRAM"
4B 20 PRINT " [DOWN2] "SPC(21) " [RV
SON] * [RVSOFF] ":PRINT "[UP]"
SPC(6) "OPTION NUMBER [RVSON
] [RVSOFF] ";
27 21 POKE204,0:GETNK$:IFNK$<"1
"ORNK$>"5"THEN21
14 22 POKE204,1:POKE207,0:PRINT
"[RVSON]"NK$:NK=VAL(NK$)
18 23 ONNKGOTO24,101,46,116,125

76 24 Q3=PEEK(0)-1:Q1=1
A6 25 IFQ1=>Q3THEN28
B3 26 IFQ1<0ANDQ3/6=INT(Q3/6)TH
ENQ1=Q3-5:GOTO28
EC 27 IFQ1<0THENDN=INT(Q3/6):Q1
=DN*6+1
B9 28 PRINT "[CLR,DOWN,RVSOFF,SP
C4]DISK NAME: [RVSON]"DT$(0)
" ":PRINT "[DOWN] ":DR=5
29 29 FORI=Q1TOQ1+DR
C3 30 IFDT$(I)=CHR$(32)+"[SPC18
]"ORDT$(I)=""THENDT$(I)="" :D
R=DR-1
9D 31 NEXT
2A 32 FORI=Q1TOQ1+DR
71 33 IFI>9THENW$=""[LEFT,RVSON]
":GOTO35
00 34 W$="" [LEFT,RVSON]"
BD 35 PRINT "[LEFT]"I:W$:LEFT$(D
T$(I),16)": "A2$:PRINT "[UP2]"
SPC(20);
23 36 PRINT "[RVSON]"J$(I)"[DOWN
2]" :NEXT
B3 37 PRINTA1$ "[DOWN3]F1-MENU F
3-ADD F5-FWD F7-BACK E-ERASE
[HOME]":POKE198,0:WAIT198,1
01 38 IFPEEK(197)=48THEN119
55 39 IFPEEK(197)=4THEN15
0A 40 IFPEEK(197)=5THEN54
65 41 IFPEEK(197)=6ANDQ1-Q3THEN
Q1=1:GOTO25
91 42 IFPEEK(197)=6ANDQ1+6>Q3TH
ENQ1=1:GOTO25
91 43 IFPEEK(197)=6ANDQ1<Q3THEN
Q1=Q1+6:GOTO25
F2 44 IFPEEK(197)=3THENQ1-Q1-6:
GOTO25
EF 45 GOTO38
D0 46 PRINT "[UP2]"SPC(28)"[LEFT
,DOWN2]":PRINT "[UP2]"SPC(9)"
SAVING "CHR$(34)"FILE DES"
;
10 47 PRINTCHR$(34)"...":FORQ=1
TOQ3:IFJ$(Q)=""THENJ$(Q)=""[R
VSON]"
03 48 NEXTQ
82 49 CLOSE1:CLOSE15:OPEN1,8,15
,"S0:FILE DES":CLOSE1:OPEN15
,8,15,"I"
7A 50 OPEN1,8,8,"0:FILE DES,S,W
":GOSUB98
72 51 PRINT#1,Q3
30 52 FORI=1TOQ3:PRINT#1,LEFT$(
DT$(I),16):PRINT#1,J$(I):NEX
T:CLOSE1
7A 53 PRINT "[UP,SPC31]":GOTO15
BB 54 PRINTA1$ "[DOWN3]"SPC(8)"[
RVSON]F3-ADD[HOME]"
81 55 PRINTA1$ "[DOWN] WHICH F
ILE ":POKE198,0:INPUTDF$:DF
=VAL(DF$)
9B 56 IFDF$="0"THENPRINTA1$ "[DO
WN,SPC18]":GOTO37
B4 57 IFDF<Q1ORDF>Q1+DRTHENPRIN
TA1$SPC(17)"[DOWN,SPC18]":GO
TO54
59 58 PRINTA1$ "[DOWN,SPC26]":IF
DF<7THENQ6=DF:GOTO61
46 59 Q6=INT(DF/6)+1:Q7=Q6*6-DF
:Q6=6-Q7
14 60 IFQ6=0THENQ6=6
E8 61 PRINT "[HOME,DOWN]":FORI=1
TOQ6:PRINT "[DOWN]":NEXT
46 62 DT$(DF)=LEFT$(DT$(DF),16)

84 63 F1$=""
5B 64 PRINT "[UP]"SPC(19)"[RVSON
]>"SPC(LEN(F1$)):CR$="<":QB
=LEN(CR$):NM=1
BC 65 PRINT "[RVSON]"MID$(CR$,NM
,2): "[LEFT]":NM=NM*(NM*QB)
+1
0E 66 GETOP$:IFOP$=""OROP$=CHR$(
34)OROP$=CHR$(44)OROP$=CHR$(
58)THEN66
76 67 IFOP$=CHR$(59)THEN66
90 68 IFOP$=CHR$(13)ANDLEN(F1$)
=18THEN84
64 69 IFLEN(F1$)=18ANDOP$<>CHR$(
20)THENF1$=F1$+OP$:PRINTOP$
"[RVSOFF,SPC4,RVSON]<[LEFT]"
":GOTO65
50 70 IFLEN(F1$)=55THENQ5=1
E6 71 IFLEN(F1$)=55ANDOP$<>CHR$(
20)ANDOP$<>CHR$(13)THEN66
4F 72 IFOP$=CHR$(13)ANDLEN(F1$)
=0THEN28
A6 73 IFOP$=CHR$(13)THEN84
66 74 IFOP$=CHR$(20)ANDLEN(F1$)
=19THEN81
A6 75 IFOP$=CHR$(20)ANDLEN(F1$)
>0THEN78
AE 76 IFASC(OP$)<32ORASC(OP$)>9
5THEN65
69 77 F1$=F1$+OP$:PRINTOP$:GOT
O65
46 78 F1$=LEFT$(F1$,LEN(F1$)-1)
B1 79 IFLEN(F1$)=54THENPRINT "[R
VSOFF] [RVSON,LEFT2]":GOTO6
5
F0 80 PRINT "[RVSON] [LEFT2]":G
OTO65
E8 81 F1$=LEFT$(F1$,LEN(F1$)-1)
:PRINT "[RVSON] ":D=19
40 82 IFQ5=1THENPRINT "[UP3]"SPC
(38)"" :GOTO65
61 83 PRINT "[UP2]"SPC(38)"" :GO
TO65
FC 84 J$(DF)=F1$
BB 85 IFLEN(J$(DF))<19THEN28
42 86 J$(DF)=MID$(J$(DF),1,19)+
"[RVSOFF,SPC4,RVSON]"+MID$(J
$(DF),20,37)
33 87 J$(DF)=J$(DF)+"[UP]":GOTO
28
38 88 OPEN15,8,15,"I"
71 89 OPEN1,8,8,"0:FILE DES,S,R
":GOSUB98:IFER=62THENCLOSE15
:RETURN
83 90 PRINT "[UP]"SPC(19)"FILE D
ES[SPC5]"
BE 91 INPUT#1,Q3
8B 92 FORI=1TOQ3:INPUT#1,K$(I):
INPUT#1,J$(I):J1$(I)=J$(I):N
EXT
C3 93 CLOSE1:CLOSE15:NF=PEEK(0)
-1
27 94 FORQ1=1TONF:IFLEFT$(DT$(Q
1),16)=K$(Q1)THENJ$(Q1)=J1$(
Q1):GOTO97
8D 95 FORQ2=1TOQ3:IFLEFT$(DT$(Q
1),16)=K$(Q2)THENJ$(Q1)=J1$(
Q2):Q2=Q3+2
3C 96 NEXTQ2:IFQ2=Q3+1THENJ$(Q1
)=""
B1 97 NEXTQ1:RETURN
1D 98 INPUT#15,ER,ER$:IFER=0THE
NRETURN
FE 99 IFER=62ANDFL=1THENFL=0:RE
TURN
2F 100 PRINT "[CLR,DOWN4]DISK ER
ROR:[RVSON]"ER,ER$:PRINT "[HO
ME,DOWN4]"SPC(14)"[RVSON] [R
VSOFF]":END
57 101 PRINT "[CLR,DOWN2,RVSON]D
ISK COMMANDS[RVSOFF]":PRINT
[DOWN2]SUMMARY-":PRINT
C2 102 PRINT "[DOWN]V0"SPC(20)"V
ALIDATE A DISK":PRINT "[DOWN]
I0"SPC(20)"INITIALISE"
3D 103 PRINT "[DOWN]S0:FILENAME"
SPC(11)"SCRATCH A FILE"
1B 104 PRINT "[DOWN]R0:NEWNAME=O
LDNAME[SPC4]RENAME A FILE"
DB 105 PRINT "[DOWN2,RVSON]OPTIO
N REQUIRED[RVSOFF] (RETURN F
OR MENU)":Q6$="" :PRINT "[DOWN
]>";
66 106 GETP1$:IFP1$=""THEN113
81 107 IFLEN(Q6$)=37ANDP1$<>CHR
$(13)ANDP1$<>CHR$(20)THEN106
5E 108 P1=ASC(P1$):IFP1=13THEN1
14
84 109 IFP1<>20THEN112
FD 110 IFLEN(Q6$)<>0THENQ6$=MID
$(Q6$,1,LEN(Q6$)-1):PRINTP1$
;
5D 111 GOTO106
B3 112 IFP1<32ORP1>90THEN106
53 113 PRINT "[LEFT]":P1$:Q6$=
Q6$+P1$:GOTO106
C1 114 IFQ6$=""THEN15
5B 115 CLOSE1:OPEN1,8,15,Q6$:CL
OSE1:CLOSE15:OPEN15,8,15,"I"
:GOSUB98:GOTO15
49 116 PRINT "[CLR,DOWN8,SPC7]IN
SERT NEW DISK IN DRIVE"
C8 117 PRINT "[DOWN2,SPC7]PRESS
ANY KEY TO CONTINUE":POKE198
,0:WAIT198,1:GETE$
69 118 PRINT "[UP,SPC13]PLEASE W
AIT[SPC8]":CLR:GOTO9
C7 119 PRINTA1$ "[DOWN3]"SPC(30)
"[RVSON]E-ERASE[HOME]"
C6 120 PRINTA1$ "[DOWN]"SPC(10)"
[SPC29]"
C5 121 POKE198,0:DF$="" :PRINTA1
$ "[DOWN] WHICH FILE":INPUTD
F$:DF=VAL(DF$)
DD 122 IFDF$="0"THENPRINTA1$ "[D
OWN,SPC14]":GOTO37
8C 123 IFDF<Q1ORDF>Q1+DRTHEN119
E8 124 J$(DF)=""[RVSON] ":GOTO28
36 125 POKE198,0:SYS198:END
FD 126 DATA169,255,133,0,165,55
,133,71,165,56,133
5C 127 DATA72,162,1,32,198,255,
166,0,232,134,0
B2 128 DATA216,56,165,71,233,20
,133,71,176,5,166
76 129 DATA72,202,134,72,32,207
,255,164,144,208
54 130 DATA22,201,34,208,245,16
0,0,32,207,255,201
54 131 DATA34,240,249,145,71,20
0,192,19,208,242
D6 132 DATA240,207,32,204,255,9
6,0,0,0

```



**By John Fletcher**







[illegible]

```

145,250,2218
78 43 DATA136,16,246,160,3,177
    250,74,145,250,136,16,248,16
    5,250,24,2296
62 44 DATA105,8,133,250,144,8,2
    30,251,165,251,201,50,240,3,
    76,40,2155
E7 45 DATA195,96,204,223,193,14
    4,244,206,218,193,240,30,96,
    20,8,0,2310

```

### LISTING 5: CHAR DEMO

```

09 100 POKES3280,3:POKES3281,3:
PRINT"[BLACK,CLR,DOWN10,RIGHT]PLEASE WAIT..READING TEXT
"
4A 110 IFPEEK(16384)-45THEN200
42 120 :
A0 130 AS="--., HELLO THERE FEL
LOW YC READERS ..-- THIS RO
UTINE WILL ALLOW YOU"
E0 140 AS=AS+" TO SCROLL TEXT W
ITHIN A CHARACTER .. AS YOU
CAN SEE SOME STUNNING"
25 150 AS=AS+" EFFECTS CAN BE A
CHIEVED WITH EASE !!! YOU CA
N SCROLL UP OR LEFT..."
9A 160 :
0B 170 L=LEN(AS):FORI=1TOL:POKE
16383+I,ASC(MID$(AS,I,1)):NE
XT:POKE16384+L,255
96 180 FORI=0T07:POKE12288+I+25
5*8,0:NEXT
84 190 :
6F 200 PRINT"[C5,CLR]":POKES328
1,0:POKES3280,0:POKE16383,0:
POKES3265,0
24 210 PRINT"[SPCB,CP25]"
39 220 PRINT"[SPCB,RUSON,C*,RUS
OFF,CP23,SN,CH]"
B6 230 PRINT"[SPCB,RUSON] [RUSO
FF,SPC4,CH] [CYAN]INVENTED B
Y[C5] [CN,SPC4,CH2]"
83 240 PRINT"[SPCB,SE,CY4,CH,RU
SON,SPC13,RUSOFF,CN,CY4,SM,C
H]"
E7 250 PRINT"[SPCB,CY,SP,CY,SO,
CY,CH,RUSON,SPC5,RUSOFF,SM,C
P,SN,RUSON,SPC5,RUSOFF,CN,CY
,SP,CY,SO,CY]"
13 260 PRINT"[SPC9,CN,RUSON,WHI
TE]A[RUSOFF,C5,CH] [CH,RUSON
,SPC5,RUSOFF,CN,RUSON,WHITE]
P[RUSOFF,C5,CH,RUSON,SPC5,RU
SOFF,CN] [CN,RUSON,WHITE]A[R
USOFF,C5,CH]"
02 270 PRINT"[SPC9,CN,RUSON,WHI
TE]B[RUSOFF,C5,CH] [CH,RUSON
,SPC5,RUSOFF,CN,RUSON,WHITE]
Q[RUSOFF,C5,CH,RUSON,SPC5,RU
SOFF,CN] [CN,RUSON,WHITE]B[R
USOFF,C5,CH]"
3E 280 PRINT"[SPCB,CP,SE,CP,SL,
CP,CH,RUSON,SPC5,RUSOFF,SN,C
Y,SM,RUSON,SPC5,RUSOFF,CN,CP
,SE,CP,SL,CP]"
05 290 PRINT"[SPCB,RUSON,C*,RUS
OFF,CP4,CH,RUSON,SPC13,RUSOF
F,CN,CP4,SN,CH]"
FA 300 PRINT"[SPCB,RUSON] [WHIT
E,SA,SB,SC,SD,RUSOFF,C5,CH,Y
ELLOW]JOHN FLETCHER[C5,CN,RU
SON,WHITE,ST,SU,SV,SW,RUSOFF
,C5,CH2]"

```

```

75 310 PRINT"[SPC8,SE,CY23,SM,CH]"
FF 320 PRINT"[SPC8,CY,SP,CY,SO,
CY6,SP,SM,CP,SN,SO,CY6,SP,CY
,SO,CY]"
F4 330 PRINT"[SPC9,CN,RVSON,WHI
TE]H[RUSOFF,C5,CH,SPC6,CN2,R
VSON,WHITE]W[RUSOFF,C5,CH2,S
PC6,CN,RVSON,WHITE]H[RUSOFF,
C5,CH]"
C7 340 PRINT"[SPC9,CN,RVSON,WHI
TE]I[RUSOFF,C5,CH,SPC6,CN2,R
VSON,WHITE]X[RUSOFF,C5,CH2,S
PC6,CN,RVSON,WHITE]I[RUSOFF,
C5,CH]"
26 350 PRINT"[SPC9,CN,RVSON,WHI
TE]J[RUSOFF,C5,CH,SPC6,CN,SN
,CY,SM,CH,SPC6,CN,RVSON,WHIT
E]J[RUSOFF,C5,CH]"
9A 360 PRINT"[SPC9,CN,RVSON,WHI
TE]K[RUSOFF,C5,CH,SPC4,RVSON
,SE,RUSOFF,SO,CY5,SP,SM,SPC4
,CN,RVSON,WHITE]K[RUSOFF,C5,
CH]"
BB 370 PRINT"[SPC9,CN,RVSON,WHI
TE]L[RUSOFF,C5,CH,SPC4,RVSON
][RUSOFF,CG,RVSON,SPC5,RUSO
FF,CN][CH,SPC3,CN,RVSON,WHI
TE]L[RUSOFF,C5,CH]"
BA 380 PRINT"[SPC9,CN,RVSON,WHI
TE]M[RUSOFF,C5,CH,SPC4,SE,CH
,RVSON,SPC5,RUSOFF,CN,SM,CH,
SPC3,CN,RVSON,WHITE]M[RUSOFF
,C5,CH]"
52 390 PRINT"[SPC7,RVSON,SE,RUS
OFF,SO,CY7][CH,RVSON,SPC5,R
USOFF,CN][CY6,SP,SM]"
6D 400 PRINT"[SPC7,RVSON][RUSO
FF,CG,RVSON,WHITE,SB,SC,SD,S
E,SF,SG,SH,SI,RUSOFF,C5,CH,R
VSON,SPC5,RUSOFF,CN,RVSON,WH
ITE,SQ,SR,SS,ST,SU,SV,SW,RVS
OFF,C5,CN][CH]"
20 410 PRINT"[SPC7,RVSON][RUSO
FF,CG,SPC7,RVSON,SE,RUSOFF,C
G,RVSON,SPC5,RUSOFF,CN,SM,SP
C6,CN][CH]"
61 420 PRINT"[SPC7,SE,CY7,SP,RV
SON][RUSOFF,SL,CPS,SO][SO,
CY6,SM,CH]"
66 430 PRINT"[SPC7,CY8,SP,SE,SP
C7,SM,SO,CY7]"
CC 440 PRINT"[SPC16,CY9]";
F9 450 :
F4 460 REM **** COPY CHARS. FRO
M ROM ****
ED 470 :
3D 480 SYS 49900
D1 490 :
16 500 REM **** RE-DESIGN CHAR.
SET ****
C5 510 :
87 520 SYS 49952
2E 530 :
92 540 REM **** SET PARMS FOR S
CROLL ****
12 550 :
2D 560 SYS 49152,16384,12288,19
2,25
3B 570 SYS 49500,16384,12288,12
8,25
7C 580 :
8F 590 REM **** INIT. INTERRUPT
S ... ****
60 600 :
DF 610 POKE53272,28:SYS 49800
2C 620 PRINT"[HOME]"

```



# TRANS-SCRIPT

*Many Plus/4 owners have upgraded to the SCRIPT-PLUS wordprocessor. Unfortunately you can't use 3+1 files with this wordprocessor. TRANS-SCRIPT changes all this.*

**H**aving recently obtained the SCRIPT/PLUS cartridge I was left with the problem of converting several disks of 3+1 wordprocessor files into a format suitable for SCRIPT/PLUS.

This program *TRANS-SCRIPT* will enable 3+1 format files to be converted using either single or twin disk units. The program is menu driven and will prompt for single/twin disk units, filename, directory or exit. Exit from the directory is by pressing return.

In single disk mode the program reads the file, converts it and then writes it back to the disk using the same filename but with the first character of the filename overwritten with an exclamation mark.

When using twin disk units the file is written to the second unit using the same filename.

The conversion is done on a character by character basis but by use of machine code the process is fairly swift, especially if 1551 disk drives are used. The layout of the original 3+1 file is preserved, but the 3+1 embedded commands are removed.

## Getting it in

To enter the program use the MONITOR and the M command to type in the hex dump listing.

Then SAVE "TRANS-SCRIPT",08,1001,14C9

The program can then be loaded and

run like a normal BASIC program.

N.B. The program must be located at the normal start of basic area. E.g. Hex 1001. GRAPHIC CLR can be used to ensure this.

## File format

The 3+1 wordprocessor file is held on disk in the following format. Byte 01 and 02 point to the end address of the document in memory. Byte 03 is the number of lines in the document. The next 99 bytes are the text pointers. This is followed by a further 77 bytes for the Tab set-

tings. Finally there is a further gap until the start of the document text. This gives a total of 303 bytes before the document text which can be discarded during the conversion process.

The document text is stored in memory and on disk in CBM screen code. The text is stored in 77 character lines. The carriage return "Hex 0D" is replaced by "Hex 9F". The line after the carriage return is padded to the full 77 characters with spaces. These spaces will be discarded during the conversion, reducing the size of the converted document.

PROGRAM: TRANS-SCRIPT

```
1001 0D 10 0A 00 9E 28 34 31:
1009 31 31 29 00 00 00 A0 09:
1011 A9 00 99 D7 00 88 D0 FA:
1019 A9 32 B5 E2 A9 03 B5 E3:
1021 A0 FF A9 93 20 D2 FF C8:
1029 B9 36 13 D0 F7 20 5A 12:
1031 A2 0B A0 02 18 20 F0 FF:
1039 A0 FF A9 53 20 D2 FF C8:
1041 B9 3B 13 D0 F7 20 83 11:
1049 AD 32 03 C9 53 D0 03 4C:
1051 6E 10 C9 54 D0 03 4C FB:
1059 10 C9 58 D0 E8 A9 1B 20:
1061 D2 FF A9 4C 20 D2 FF A9:
1069 09 20 D2 FF 60 AD 32 03:
1071 B5 E0 20 5A 12 A2 06 A0:
1079 02 18 20 F0 FF A0 FF A9:
1081 53 20 D2 FF C8 B9 8C 13:
1089 D0 F7 20 83 11 A5 DE C9:
1091 01 D0 14 AD 32 03 C9 24:
1099 D0 06 20 9F 11 4C 73 10:
```

```
10A1 C9 5B D0 E6 4C 2E 10 A9:
10A9 02 A2 0B A0 02 B6 E1 20:
10B1 BA FF 20 68 12 A5 DE A2:
10B9 32 A0 03 20 BD FF 20 C0:
10C1 FF 90 03 4C B9 11 20 FC:
10C9 11 B0 A7 A9 21 BD 32 03:
10D1 A9 03 A2 0B A0 03 B6 E1:
10D9 20 BA FF 20 7F 12 A5 DE:
10E1 A2 32 A0 03 20 BD FF 20:
10E9 C0 FF 90 03 4C B9 11 20:
10F1 FC 11 B0 03 20 93 12 4C:
10F9 73 10 AD 32 03 B5 E0 20:
1101 5A 12 A2 06 A0 02 18 20:
1109 F0 FF A0 FF A9 54 20 D2:
1111 FF C8 B9 EE 13 D0 F7 20:
1119 B3 11 A5 DE C9 01 D0 14:
1121 AD 32 03 C9 24 D0 06 20:
1129 9F 11 4C 00 11 C9 58 D0:
1131 E6 4C 2E 10 A9 02 A2 0B:
1139 A0 02 B6 E1 20 BA FF 20:
1141 68 12 A5 DE A2 32 A0 03:
1149 20 BD FF 20 C0 FF 90 03:
1151 4C B9 11 20 FC 11 B0 A7:
1159 A9 03 A2 09 A0 03 B6 E1:
1161 20 BA FF 20 7F 12 A5 DE:
```



```

1169 A2 32 A0 03 20 8D FF 20:
1171 C0 FF 90 03 4C B9 11 20:
1179 FC 11 B0 83 20 93 12 4C:
1181 00 11 A2 14 A0 12 18 20:
1189 F0 FF A0 00 20 CF FF C9:
1191 0D F0 08 99 32 03 C8 C0:
1199 10 D0 F1 84 DE 60 A9 93:
11A1 20 D2 FF A9 18 20 D2 FF:
11A9 A9 4C 20 D2 FF 20 8C C8:
11B1 20 E4 FF C9 0D D0 F9 60:
11B9 85 DF A0 0C B9 84 14 99:
11C1 D9 0F 88 D0 F7 A5 DF 69:
11C9 2F 8D E6 0F 20 48 12 A5:
11D1 E0 C9 53 D0 03 4C 73 10:
11D9 4C 00 11 20 CC FF A2 14:
11E1 A0 02 18 20 F0 FF A0 FF:
11E9 A9 50 20 D2 FF C8 B9 92:
11F1 14 D0 F7 20 E4 FF C9 0D:
11F9 D0 F9 60 A9 0F A8 A6 E1:
1201 20 BA 00 20 2C 2E 24 FF:
1209 FF FF FF FF 0F 20 C6 C9:
1211 14 CF FF C9 30 D0 11 20:
1219 CF FF C9 30 D0 0D A9 0F:
1221 20 C3 FF 20 CC FF 18 60:
1229 20 CF FF 20 CF FF A2 18:
1231 A0 00 18 20 F0 FF 20 CF:
1239 FF C9 0D F0 05 20 D2 FF:
1241 D0 F4 A9 0F 20 C3 FF A9:
1249 03 20 C3 FF A9 02 20 C3:
1251 FF 20 CC FF 20 DC 11 38:
1259 60 A0 FF A9 93 20 D2 FF:
1261 C8 B9 75 14 D0 F7 60 A2:
1269 00 A4 DE BD C1 14 91 E2:
1271 C8 E8 E0 04 D0 F5 18 A5:
1279 DE 69 04 85 DE 60 38 A5:
1281 DE E9 04 A8 A2 00 BD C5:

```

```

1289 14 91 E2 C8 E8 E0 04 D0:
1291 F5 60 A2 02 20 C6 FF A2:
1299 FF A0 30 20 E4 FF CA D0:
12A1 FA 20 E4 FF 88 D0 FA 20:
12A9 CC FF A2 00 86 DA 86 DB:
12B1 E8 86 D9 A2 02 20 C6 FF:
12B9 20 E4 FF 85 DC 20 87 FF:
12C1 48 20 CC FF E6 DB A5 DB:
12C9 C9 4D 00 08 A2 00 86 DB:
12D1 A2 01 86 DA A5 DC C9 9F:
12D9 D0 14 A2 03 20 C9 FF A9:
12E1 0D 20 D2 FF 20 CC FF A2:
12E9 00 86 D9 4C 18 13 29 80:
12F1 30 25 A5 DC 85 DB 29 3F:
12F9 06 D8 24 D8 10 02 09 80:
1301 70 02 09 40 85 DC A5 D9:
1309 F0 0D A2 03 20 C9 FF A5:
1311 DC 20 D2 FF 20 CC FF A5:
1319 DA F0 07 A2 00 86 DA E8:
1321 86 D9 68 29 40 F0 8C A9:
1329 03 20 C3 FF A9 02 20 C3:
1331 FF 20 E7 FF 60 93 08 18:
1339 4D 00 20 3D 20 53 49 4E:
1341 47 4C 45 20 44 49 53 48:
1349 20 44 52 49 56 45 0D 0D:
1351 20 20 54 20 3D 20 54 57:
1359 49 4E 20 44 49 53 48 20:
1361 44 52 49 56 45 53 0D 0D:
1369 20 20 58 20 3D 20 45 58:
1371 49 54 0D 0D 0D 0D 0D 20:
1379 20 20 20 45 4E 54 45 52:
1381 20 4F 50 54 49 4F 4E 3F:
1389 20 53 00 49 4E 47 4C 45:
1391 20 44 49 53 48 20 4F 50:
1399 45 52 41 54 49 4F 4E 0D:
13A1 20 20 20 20 20 20 20:

```

```

13A9 28 4F 4E 20 55 4E 49 54:
13B1 20 38 29 0D 0D 0D 20 20:
13B9 24 20 3D 20 44 49 52 45:
13C1 43 54 4F 52 59 0D 0D 20:
13C9 20 58 20 3D 20 45 58 49:
13D1 54 0D 0D 0D 0D 0D 0D 0D:
13D9 0D 20 20 45 4E 54 45 52:
13E1 20 46 49 4C 45 4E 41 4D:
13E9 45 3F 20 2A 00 57 49 4E:
13F1 20 44 49 53 48 20 4F 50:
13F9 45 52 41 54 49 4F 4E 8D:
1401 0D 20 20 50 4C 41 43 45:
1409 20 53 4F 55 52 43 45 20:
1411 44 49 53 48 20 49 4E 20:
1419 55 4E 49 54 20 38 0D 20:
1421 20 26 20 54 48 45 20 54:
1429 41 52 47 45 54 20 44 49:
1431 53 48 20 49 4E 20 55 4E:
1439 49 54 20 39 0D 0D 20 20:
1441 24 20 3D 20 44 49 52 45:
1449 43 54 4F 52 59 0D 0D 20:
1451 20 58 20 3D 20 45 58 49:
1459 54 0D 0D 0D 0D 0D 0D 0D:
1461 20 20 45 4E 54 45 52 20:
1469 46 49 4C 45 4E 41 4D 45:
1471 3F 20 2A 00 93 18 4D 20:
1479 20 20 20 20 20 20 20 20:
1481 20 20 20 20 54 52 41 4E:
1489 53 2D 53 43 52 49 50 54:
1491 00 52 45 53 53 20 52 45:
1499 54 55 52 4E 20 54 4F 20:
14A1 43 4F 4E 54 49 4E 55 45:
14A9 20 20 20 20 20 20 20 20:
14B1 20 20 20 00 09 2F 0F 20:
14B9 05 12 12 0F 12 20 23 20:
14C1 2C 53 2C 52 2C 53 2C 57:

```

## LIFESAVERS

## C64

## MACHINE CODE SAVE

1/1

For those of you that do not possess a monitor or assembler, saving areas of memory is somewhat difficult and time consuming. This small programme will enable you to SAVE any area of memory you like, except for RAM hidden under the ROMs.

The syntax for using the routine is as follows:

```
SYS50000, "name", 8, 1, SA, EA+1
```

Where EA and SA are the end and start address respectively.

P.A.Eves

```

1 X=50000
2 READ2: IF2=256 THEN END
3 POKE X, 2: X=X+1: GOTO 2
4 DATA 32, 253, 174, 32, 212, 225, 32, 2
53, 174, 32, 138, 173, 32, 247, 183, 165
, 20, 72, 165, 21
5 DATA 72, 32, 253, 174, 32, 138, 173, 3
2, 247, 183, 166, 20, 164, 21, 104, 133,
252, 104, 133
6 DATA 251, 169, 251, 76, 95, 225, 256

```



# Extended Basic For The Commodore Plus/4

*Although printers and disk drives are optional, they are necessary if full use is to be made of the existing Basic.*

**T**his program adds an extra 39 commands to the existing Basic. Some commands are intended for disk users only and some are intended for printer users only (primarily MPS803 users but these commands may be compatible with other printers i.e. MPS801). The other commands are intended to aid basic programming, i.e. an old command which when executed will regain a 'newed' program.

The program sits from \$1000 to \$2100 with Basic starting at \$2101. The graphic screen can still be used, as the program will automatically relocate itself to \$4000 when you turn on the graphic screen for the first time. The program, once relocated to \$4000 by the initialisation of the hi-res screen, will automatically relocate itself back to \$1000 once graphicclr has been activated.

The program uses \$0600 to \$068D for the relocating routine and the zero page is used by some of the commands.

## Typing It In

Type in listing one and save it to disk or tape, run the program and if any data errors are detected the program will automatically list the line in which the error occurred. Therefore editing can be carried out to correct the line.

The program can then be re-saved

and re-executed. Once the data has been read into memory the computer will prompt you with the device (tape or disk). If you are using fast loader from February's issue or a different device number for the disk drive, you will need to change the device number to the corresponding device in line 90.

When the device has been entered the computer will then save the machine-code program produced to the chosen device under the name "extended Basic". If you are using the fast loader program, delete line 10 in listing one before saving it and type in the following basic line:—  
POKE 44, 64: POKE 43,1: POKEDC ("4000), 0: NEW

Load in the 'fast loader' program, adding the lines for relocating it from August's issue. Run the program and relocate fast loader to \$3D00 (remember delete line 1165 in the fast loader relocate program first); type new and load in listing one and add the following line:—  
95 IFAS="T"THENSYSDECK("3D00")

## Execute The Program

Listings two to six are demo programs and therefore need not be entered unless you want a demonstration of the commands at work (N.B. the extended Basic must be executed before the demo programs are entered. The demo programs must be

entered in order, and saved before the next one is entered. If you haven't a disk drive you need not type in listing six.) When the demo programs have been entered and saved listing two can then be loaded by using the chain command.

## Commands Summary and Format

### COMMAND: APPEND

FORMAT: APPEND["FILENAME"]  
[DEVICE]]

Abbreviated to A(SHIFT)P

AFFECTED BASIC ABBREVIATIONS:  
None

MODES: Direct and Program

RECOMMENDED MODE: Direct

PURPOSE: To append a program from tape or disk to the end of the current program in memory.

### COMMAND: BSAVE

FORMAT: BSAVE["FILENAME"]  
[DEVICE], EOT FLAG[, START[,  
FINISH]]]]]

Abbreviated to B(SHIFT)S

AFFECTED BASIC ABBRECIATIONS:  
None

MODES: Direct and program

RECOMMENDED MODE: Either

PURPOSE: To save a block of memory to tape or disk. The start address defaults to the start of the basic and the end address defaults to the end of the current



basic program. The EOT flag is the same as for the basic SAVE command.

**COMMAND: CGOSUB**

FORMAT: CGOSUB Variable Expression.

Abbreviated to CGO(SHIFT)S

AFFECTED BASIC ABBREVIATIONS: None

MODES: Direct and Program

RECOMMENDED MODE: Either

PURPOSE: To GOSUB a line number evaluated from the variable expression.

**COMMAND: CGOTO**

FORMAT: CGOTO Variable Expression.

Abbreviated to C(SHIFT)G

AFFECTED BASIC ABBREVIATIONS: None

MODES: Direct and Program

RECOMMENDED MODE: Either

PURPOSE: To GOTO a line number evaluated from the variable expression.

**COMMAND: CHAIN**

FORMAT: CHAIN["FILENAME"][, DEVICE[, RELOCATE FLAG]]

Abbreviated to C(SHIFT)H

AFFECTED BASIC ABBREVIATIONS:

CHR\$ —> CH(SHIFT)R

MODES: Direct and Program

RECOMMENDED MODE: Direct

PURPOSE: To load in a program whilst keeping the current variables intact.

**COMMAND: DISK**

FORMAT: DISK STRING[, DEVICE(8-II)]

Abbreviated to D(SHIFT)I

AFFECTED BASIC ABBREVIATIONS: DIM

MODES: Direct and Program

RECOMMENDED MODE: Either

PURPOSE: To send a command to the disk drive.

**COMMAND: DOKE**

FORMAT: DOKE ADDRESS, VALUE(0-65535)

Abbreviated to D(SHIFT)O

AFFECTED BASIC ABBREVIATIONS: None

MODES: Direct and Program

RECOMMENDED MODE: Either

PURPOSE: To POKE a 16 bit number in (LO/HI) format in locations address AND address+1.

**COMMAND: OMERGE**

FORMAT: DMERGE"FILENAME"C, DEVICE(8-II)

Abbreviated to D(SHIFT)M

AFFECTED BASIC ABBREVIATIONS: None

MODES: Direct and Program

RECOMMENDED MODE: Direct

PURPOSE: To merge a program from disk to the one currently in memory.

**COMMAND: DPROC**

FORMAT: DPROC NAME[(VARIABLES)]

Abbreviated to D(SHIFT)P

AFFECTED BASIC ABBREVIATIONS: None

MODES: Direct and Program

RECOMMENDED MODE: Program

PURPOSE: To define a procedure under the name-NAME.

**COMMAND: DUMP**

FORMAT: DUMP FLAG (0 OR 1)

Abbreviated to D(SHIFT)U

AFFECTED BASIC ABBREVIATIONS: None

MODES: Direct and Program

RECOMMENDED MODE: Either

PURPOSE: To dump the Hi-res or Lo-res screen to printer. If flag equals nought then the Lo-res screen will be dumped else the Hi-res screen will be dumped to printer.

**COMMAND: ENTER**

FORMAT: ENTER(X CO-ORDINATE, Y CO-ORDINATE)[STRING]

ABBREVIATED TO E(SHIFT)N

AFFECTED BASIC ABBREVIATIONS: END

MODES: Program

PURPOSE: To input variables at a specific location on the screen.

**COMMAND: EPROC**

FORMAT: EPROC

ABBREVIATED TO E(SHIFT)P

AFFECTED BASIC ABBREVIATIONS: None

MODES: DIRECT AND PROGRAM

RECOMMENDED MODE: Program

PURPOSE: To return from a procedure. If this is not executed in a proc routine then a RETURN WITHOUT GOSUB ERROR will occur.

**COMMAND: FAST**

FORMAT: FAST

ABBREVIATED TO F(SHIFT)A

AFFECTED BASIC ABBREVIATIONS:

None

MODES: Direct and Program

RECOMMENDED MODE: Either

PURPOSE: To switch out the screen and therefore speed up basic by approximately 30%

**COMMAND: FIND**

FORMAT: FIND[TEXT]

ABBREVIATED TO F(SHIFT)I

AFFECTED BASIC ABBREVIATIONS: None

MODES: Direct

PURPOSE: To list the lines where the text occurs

**COMMAND: HIMEM**

FORMAT: HIMEM ADDRESS

ABBREVIATED TO H(SHIFT)I

AFFECTED BASIC ABBREVIATIONS: None

MODES: Direct or Program

RECOMMENDED MODE: Either

PURPOSE: To set the bottom of memory to the address specified.

**COMMAND: LOMEM**

FORMAT: LOMEM ADDRESS

ABBREVIATED TO L(SHIFT)O

AFFECTED BASIC ABBREVIATIONS: LOAD —> LO(SHIFT)A

MODES: Direct or Program

RECOMMENDED MODE: Direct

PURPOSE: To set the bottom of memory to the address specified. This will perform a NEW at the new location when executed.

**COMMAND: MERGE**

FORMAT: MERGE["FILENAME"][, DEVICE]]

ABBREVIATED TO M(SHIFT)E

AFFECTED BASIC ABBREVIATIONS: None

MODES: Direct or Program

RECOMMENDED MODE: Direct

PURPOSE: To merge a program from disk or tape to the one currently in memory.

**COMMAND: OLD**

FORMAT: OLD

ABBREVIATED TO O(SHIFT)L

AFFECTED BASIC ABBREVIATIONS: None

MODES: Direct or Program

RECOMMENDED MODE: DIRECT

PURPOSE: To regain back a NEW ed program.



**COMMAND: PLIST**

FORMAT: PLIST[FIRST LINE ]—  
[LAST LINE ]]

ABBREVIATED TO PL(SHIFT)I

AFFECTED BASIC ABBREVIATIONS:  
None

MODES: Direct or Program

RECOMMENDED MODE: Either

PURPOSE: To list a program from  
memory to printer.

**COMMAND: PLOT**

FORMAT: PLOT(X CO-ORDINATE, Y  
CO-ORDINATE)

ABBREVIATED TO P(SHIFT)L

AFFECTED BASIC ABBREVIATIONS:  
None

MODES: Direct or Program

RECOMMENDED MODE: Either

PURPOSE: To position the cursor in the  
Lo-res screen at the specified  
co-ordinates.

**COMMAND: POP**

FORMAT: POP

ABBREVIATED TO P(SHIFT)O

AFFECTED BASIC ABBREVIATIONS:

POKE → PO(SHIFT)K

MODES: Program

PURPOSE: To enable you to RETURN  
from a GOSUB, CGOSUB or PROC  
routine as it takes the GOSUB parameters  
off the stack, i.e. it performs a RETURN  
without returning to the original location.  
If this is not used in a CGOSUB, GOSUB  
or PROC routine then a RETURN  
WITHOUT GOSUB ERROR will occur.

**COMMAND: PROC**

FORMAT: PROC NAME[  
(PARAMETERS)]

ABBREVIATED TO P(SHIFT)R

AFFECTED BASIC ABBREVIATIONS:

PRINT → PR(SHIFT)I

MODES: Direct or Program

RECOMMENDED MODE: Program

PURPOSE: To GOSUB a defined pro-  
cedure under the name-NAME.

The parameters will past into the variable  
names in the OPROC command.

**COMMAND: QUIT**

FORMAT: QUIT

ABBREVIATED TO Q(SHIFT)U

AFFECTED BASIC ABBREVIATIONS:  
None

MODES: Direct or Program

RECOMMENDED MODE: Direct

PURPOSE: To return to Basic.

**COMMAND: RECORD**

FORMAT: RECORD FILE ,  
RECORD , [OFFSET]

ABBREVIATED TO R(SHIFT)E

AFFECTED BASIC ABBREVIATIONS:

RED → RE(SHIFT)A

MODES: Direct or Program

RECOMMENDED MODE: Program

PURPOSE: To position the record  
pointer to the record number of the file  
number with offset—OFFSET for a rela-  
tive file. The equivalent disk command  
is:—

PRINT#15;"P"+CHR\$(FILE +96)+  
CHR\$(RECORD LO-BYTE)+CHR\$(  
RECORD HI-BYTE)+CHR\$(OFF-  
SET); where 15 is the file opened for the  
command channel.

NOTE. The command channel must be  
opened beforehand.

**COMMAND: SLOW**

FORMAT: SLOW

ABBREVIATED TO S(SHIFT)L

AFFECTED BASIC ABBREVIATIONS:  
None

MODES: Direct or Program

RECOMMENDED MODE: Either

PURPOSE: To restore the screen back to  
normal and hence the speed of basic after  
a FAST command has been executed.

**COMMAND: VARLIST**

FORMAT: VARLIST

ABBREVIATED TO V(SHIFT)A

AFFECTED BASIC ABBREVIATIONS:  
None

MODES: Direct

PURPOSE: To print all variables (except  
arrays and functions) from memory to the  
current output device.

**COMMAND: WINDOW**

FORMAT: WINDOW TOP, LEFT,  
RIGHT, BOTTOM

ABBREVIATED TO W(SHIFT)I

AFFECTED BASIC ABBREVIATIONS:  
None

MODES: Direct or Program

RECOMMENDED MODE: Either

PURPOSE: To set the window size.

**COMMAND: WRITE**

FORMAT: WRITE(X CO-ORDINATE,  
Y CO-ORDINATE) PRINTLIST

ABBREVIATED TO W(SHIFT)R

AFFECTED BASIC ABBREVIATIONS:  
None

MODES: Direct or Program

RECOMMENDED MODE: Either  
PURPOSE: To print at the specified co-  
ordinates on the screen.

**Functions**

FUNCTION: ACS

FORMAT: ACS(X)

ABBREVIATED TO A(SHIFT)C

AFFECTED BASIC ABBREVIATIONS:  
None

PURPOSE: To return the ARCCOS of X,  
where  $-1 \leq X \leq 1$

FUNCTION: ASN

FORMAT: ASN(X)

ABBREVIATED TO A(SHIFT)S

AFFECTED BASIC ABBREVIATIONS:  
ASC

PURPOSE: To return the ARCSIN of X,  
where  $-1 \leq X \leq 1$

FUNCTION: BIN\$

FORMAT: BIN\$(X)

ABBREVIATED TO B(SHIFT)I

AFFECTED BASIC ABBREVIATIONS:  
None

PURPOSE: To return as a string the  
binary equivalent of X, where X is an  
integer such that  $0 \leq X \leq 65535$ .

FUNCTION: BTOH\$

FORMAT: BTOH\$(X\$)

ABBREVIATED TO B(SHIFT)T

AFFECTED BASIC ABBREVIATIONS:  
None

PURPOSE: To return as a string the hexa-  
decimal value of the binary expression  
given by X\$

FUNCTION: DEEK

FORMAT: DEEK(X)

ABBREVIATED TO D(SHIFT)E

AFFECTED BASIC ABBREVIATIONS:

DEF FN → DE(SHIFT)F

PURPOSE: To return a sixteen bit  
number stored as Lo-hi format in the  
addresses X and X+1.

FUNCTION: DEG

FORMAT: DEG(X)

PURPOSE: To convert radians to degrees  
where X is the number to be converted.

FUNCTION: EVAL

FORMAT: EVAL(X\$)

ABBREVIATED TO E(SHIFT)V

AFFECTED BASIC ABBREVIATIONS:  
None



**PURPOSE:** To evaluate to a floating point number the expression in X\$. X\$ must contain functions only.

**FUNCTION:** HTOBS

**FORMAT:** HTOBS(X\$)

**ABBREVIATED TO** H(SHIFT)T

**AFFECTED BASIC ABBREVIATIONS:**

None

**PURPOSE:** To return as a string in a binary form the hexadecimal value in X\$.

**FUNCTION:** RAD

**FORMAT:** RAD(X)

**ABBREVIATED TO** R(SHIFT)A

**AFFECTED BASIC ABBREVIATIONS:**

None

**PURPOSE:** To convert degrees to radians where X is the number to be converted.

**FUNCTION:** TWO

**FORMAT:** TWO(X\$)

**ABBREVIATED TO** T(SHIFT)W

**AFFECTED BASIC ABBREVIATIONS:**

None

**PURPOSE:** To convert a binary string (X\$) to a floating point number.

PROGRAM: +4 EX BASIC 1

```
10 GRAPHIC1,1:GRAPHIC0
20 L=120:AD=4096:PRINT"[CLEAR][DOWN][DOWN]WORKING: ";
30 CK=0:FORX=0TO21:READMS:M=DEC(M$):POKEAD+X,M:CK=CK+M
40 NEXTX:READCH$:IFDEC(CH$)<>CKT:HENPRINT"ERROR IN":L:GOTO70
50 AD=AD+22:IFAD>8602THENPRINT:PRINT"[DOWN]OK...":GOTO80
60 PRINT".":L=L+10:GOTO30
70 PRINT"LCs I":L:POKE239,4:POKE1319,145:POKE1320,145:POKE1321,145:POKE1322,13:END
80 PRINT"[DOWN][DOWN]DEVICE (CRUSON)[CRUSOFF]APE OR [CRUSON][CRUSOFF]ISK)? ":DO:GETD$:LOOPUNTIL D$="D"ORD$="I":PRINTD$
85 POKEDEC("1369"),85:POKEDEC("1368"),16:POKEDEC("1372"),76:POKEDEC("2109"),144
90 DE=1:IFD$="D"THENDE=8
100 SYS8020"EXTENDED BASIC",DE,1,4097,8602
110 END
120 DATA00,00,10,00,00,9E,34,31,31,32,3A,A2,00,00,00,00,4C,E8,10,20,73,04,043A
130 DATAF0,07,C9,DE,00,03,4C,22,10,4C,D9,8B,20,73,04,C9,9C,F0,1B,A5,75,F0,0AAD
140 DATA03,4C,3B,10,A9,10,20,00,06,20,3C,C6,20,79,04,20,D1,C5,4C
```

```
,DC,8B,A5,0746
150 DATA75,F0,03,20,4B,10,4C,DC,8B,A9,40,20,00,06,A9,9C,4C,C3,C5,A2,FF,86,09E5
160 DATA3A,20,5A,8B,86,3B,84,3C,20,73,04,AA,F0,EF,90,0D,20,53,89,20,79,04,0813
170 DATAC9,DE,F0,06,4C,D9,8B,4C,2E,87,4C,22,10,A9,AF,A0,11,84,23,85,22,A0,09C3
180 DATA00,84,0B,8B,C8,20,A5,04,3B,F1,22,F0,F7,C9,80,F0,22,B1,22,30,03,C8,0A03
190 DATAD0,F9,C8,E6,0B,1B,9B,65,22,85,22,90,02,E6,23,1B,A0,00,B1,22,D0,D9,0A2F
200 DATA3B,20,79,04,4C,6A,89,05,0B,C8,4C,D4,89,AA,A0,11,84,23,A0,AF,84,22,088C
210 DATA4C,9E,8B,C9,80,90,1B,C9,9C,90,04,C9,A6,80,10,3B,E9,80,0A,AB,89,5C,0AF6
220 DATA12,4B,89,5D,12,4B,4C,73,04,4C,A1,94,A0,10,A9,7B,8C,0D,03,8D,0C,03,071A
230 DATAA0,10,A9,BD,8C,0F,03,8D,0E,03,A0,10,A9,C9,8C,11,03,8D,10,03,A9,39,0796
240 DATAA0,11,8C,0B,03,8D,0A,03,A9,13,A0,10,8D,0B,03,8C,09,03,A0,10,A9,55,062F
250 DATA8D,02,03,8C,03,03,A9,6A,A0,11,20,FC,20,A9,01,A0,21,84,2C,85,2B,A9,0798
260 DATA00,8D,00,21,60,A9,00,85,0D,20,73,04,C9,FE,F0,06,20,79,04,4C,1E,94,0738
270 DATA20,73,04,C9,9C,90,5B,C9,A6,80,57,3B,E9,80,0A,AB,89,5C,12,85,56,89,0A6B
280 DATA5D,12,85,55,20,54,00,4C,17,93,93,45,5B,54,45,4E,44,45,44,20,42,41,063A
290 DATA53,49,43,20,46,4F,52,20,54,4B,45,20,43,4F,4D,4D,4F,44,4F,52,45,20,05CC
300 DATA50,4C,55,53,2F,34,0D,42,59,20,52,4F,42,45,52,54,20,4A,41,59,43,4F,05D3
310 DATA43,4B,53,20,20,31,39,3B,36,00,4C,A1,94,4F,4C,C4,46,41,53,04,53,4C,0726
320 DATA4F,D7,57,49,4E,44,4F,D7,4C,4F,4D,45,CD,4B,49,4D,45,CD,4D,45,52,47,0893
330 DATACS,41,50,50,45,4E,C4,50,4C,4F,D4,57,52,49,54,C5,45,4E,54,45,D2,44,0909
340 DATA4F,4B,C5,51,55,49,D4,50,52,4F,C3,44,50,52,4F,C3,45,50,52,4F,C3,43,090A
350 DATA47,4F,54,CF,43,47,4F,53,55,C2,50,4F,D0,43,4B,41,49,CE,50,4C,49,53,0886
360 DATAD4,56,41,52,4C,49,53,D4,44,55,4D,D0,44,4D,45,52,47,C5,46,49,4E,C4,0904
370 DATA52,45,43,4F,52,44,A3,42,53,41,56,C5,44,49,53,CB,44,45,45,CB,54,57,0842
380 DATACF,42,49,4E,A4,42,54,4F,4B,A4,4B,54,4F,42,A4,45,56,41,CC,41,53,CE,08F8
390 DATA41,43,D3,44,45,C7,52,41,C4,00,12,84,12,AB,12,81,1B,73,12,CD,12,EA,08A7
400 DATA12,FC,13,E5,14,2B,14,4B,
```

```
14,51,14,01,14,57,14,5D,8E,0A,8D,82,15,0D,05C0
410 DATA15,16,15,3D,15,5B,1A,53,16,6B,1A,85,1B,D3,1E,07,1E,F7,1F,53,1F,9E,05CE
420 DATA17,7F,1B,8B,19,03,19,3C,19,4B,19,66,19,D5,1A,05,1A,3D,1A,1C,00,AD,04FE
430 DATA06,FF,29,EF,8D,06,FF,60,4C,7B,E3,A9,FF,A0,01,91,2B,20,1B,8B,A5,22,0A42
440 DATA1B,69,02,85,2D,A5,23,69,00,85,2E,4C,9A,8A,20,14,93,20,E4,9D,A4,15,07AA
450 DATAA5,14,84,2C,85,2B,A9,00,A0,00,91,2B,E6,2B,D0,02,E6,2C,4C,7B,8A,20,08B4
460 DATA14,93,20,E4,9D,A4,15,A5,14,84,3B,85,37,60,4C,21,99,20,D5,13,86,2B,0851
470 DATA84,2C,A0,00,9B,91,2B,20,75,13,86,2D,86,2E,20,1B,8B,20,82,13,A9,5E,075F
480 DATAA2,13,8D,02,03,8E,03,03,A9,01,85,3C,A9,FF,85,3B,A0,00,A9,D0,20,94,087B
490 DATA04,85,D2,C8,A9,D0,20,94,04,85,D3,F0,2D,C8,A9,D0,20,94,04,85,14,CB,0823
500 DATAA9,D0,20,94,04,85,15,A2,00,EB,C8,A9,D0,20,94,04,9D,FE,01,D0,F4,8A,083B
510 DATAA8,20,34,87,A5,D2,A6,D3,85,D0,86,D1,D0,8B,A9,12,A2,87,8D,02,03,8E,08AB
520 DATA03,03,20,03,87,20,C3,13,A9,00,85,AD,A6,2B,A4,2C,20,D5,FF,B0,10,20,07F6
530 DATAB7,FF,29,BF,F0,0B,20,9D,13,A2,1D,4C,83,86,60,4B,20,9D,13,6B,4C,7D,0923
540 DATAA7,A5,D0,3B,E9,02,85,14,A5,D1,E9,00,85,D1,A9,00,AB,91,14,CB,91,14,0AF0
550 DATAA5,D2,A6,D3,85,2B,86,2C,A5,D0,A6,D1,85,2D,86,2E,60,A9,00,20,8D,FF,08B9
560 DATAA2,01,A0,00,20,8A,FF,20,9D,AB,4C,7A,AB,A5,2B,85,D2,A5,2C,85,D3,A6,0AEE
570 DATA2D,A4,2E,86,D0,84,D1,60,20,D5,13,8A,3B,E9,02,85,2B,9B,E9,00,85,2C,09A1
580 DATA20,75,13,86,D0,84,D1,20,82,13,20,1B,8B,60,20,14,93,20,E4,9D,A5,14,0879
590 DATA85,F1,A5,15,85,F2,20,91,94,20,14,93,20,E4,9D,A6,15,A5,14,A0,00,91,09F9
600 DATAF1,CB,8A,91,F1,60,4C,21,99,20,8E,94,20,84,9D,E0,2B,90,03,4C,1C,99,0A4A
610 DATA8A,4B,20,A5,AB,20,84,9D,E0,19,80,F1,6B,AB,1B,20,F0,FF,20,8B,94,60,0AF0
620 DATA20,29,14,4C,00,90,20,29,14,4C,0B,91,20,17,81,4C,6B,C5,A2,00,C6,3B,0652
630 DATAB0,02,C6,3C,20,73,04,F0,10,C9,2B,D0,06,20,A1,17,4C,7B,14,9D,00,02,0764
640 DATAEB,D0,EB,A9,00,9D,00,02,A0,05,20,05,89,8B,A5,3C,91,7C,8B,A5,3B,91,09AD
650 DATA7C,8B,A5,3A,91,7C,8B,A5,39,91,7C,8B,A9,8D,91,7C,A5,2B,85,D0,A5,2C,0824
```



```

660 DATA85,D1,A0,00,A9,D0,20,94,
04,85,D2,C8,A9,D0,20,94,04,85,D3,
D0,05,A2,0B46
670 DATA11,4C,83,86,A0,04,A9,D0,
20,94,04,C9,FE,D0,0A,C8,A9,D0,20,
94,04,C9,0A9E
680 DATA8E,F0,0A,A5,D2,85,D0,A5,
D3,85,D1,D0,C9,A2,FF,E8,C8,A9,D0,
20,94,04,0E3D
690 DATAF0,11,C9,28,D0,06,20,C0,
17,4C,FE,14,DD,00,02,F0,E8,D0,DA,
DD,00,02,0ASD
700 DATAD0,D5,38,A5,D0,E9,01,85,
3B,A5,D1,E9,00,85,3C,4C,DC,8B,20,
14,93,20,0AB6
710 DATAE4,9D,4C,50,8D,A0,05,20,
05,89,8B,A5,3C,91,7C,8B,A5,3B,91,
7C,8B,A5,0A15
720 DATA3A,91,7C,8B,A5,39,91,7C,
8B,A9,8D,91,7C,20,79,04,20,0E,15,
4C,DC,8B,0918
730 DATAA9,FF,85,4A,A9,8D,85,02,
20,71,8B,F0,05,A2,0C,4C,83,86,20,
69,A7,A0,0A15
740 DATA05,20,72,A7,60,A5,32,C5,
2E,D0,09,A5,31,C5,2D,D0,03,A9,80,
2C,A9,00,0BDA
750 DATA85,0C,20,79,04,4C,80,15,
A5,0C,10,04,C6,30,C6,32,20,20,8D,
4C,DC,8B,0742
760 DATA20,6B,AB,EA,EA,EA,EA,20,
54,A9,A5,2D,85,5F,A5,2E,85,60,3B,
A5,31,85,0AF9
770 DATAS4,ES,2F,85,D2,A5,32,85,
5B,ES,30,85,D3,A5,33,3B,ES,01,85,
5B,A5,34,0A99
780 DATAE9,00,85,59,20,C7,8B,A5,
37,85,41,A5,3B,85,42,A5,5B,85,D0,
85,37,E6,0A70
790 DATAS9,A5,59,85,D1,85,3B,A6,
2B,A4,2C,A9,00,20,D5,FF,90,03,4C,
7D,A7,20,09CB
800 DATAB7,FF,29,BF,F0,05,A2,1D,
4C,83,86,86,2D,84,2E,86,5F,84,60,
A5,D0,85,0ACF
810 DATAS4,A5,D1,85,5B,3B,A5,33,
E9,01,AB,A5,34,E9,00,AA,9B,3B,ES,
5A,85,5B,0AAA
820 DATA8B,8A,ES,5B,AA,E8,9B,F0,
1F,A5,5A,1B,65,5B,85,5A,90,03,E6,
5B,1B,A5,0AEC
830 DATASF,65,5B,85,5F,90,02,E6,
60,9B,49,FF,AB,C8,C6,5B,C6,60,A9,
5A,20,94,0B26
840 DATA04,91,5F,C8,D0,F6,E6,5B,
E6,60,CA,D0,EF,3B,A5,5F,85,31,ES,
D2,85,2F,0CEF
850 DATAA5,60,85,32,ES,D3,85,30,
A5,41,85,37,A5,42,85,3B,6B,6B,20,
1B,8B,A9,09AB
860 DATA00,20,90,FF,20,E7,FF,20,
D5,8A,20,EE,8A,4C,72,15,A5,9A,C9,
80,F0,01,0B18
870 DATA60,A5,2D,85,AF,A5,2E,85,
B0,A5,B0,C5,30,D0,07,A5,AF,C5,2F,
D0,01,60,0B0B
880 DATAA0,00,A9,AF,20,94,04,C9,
B0,B0,6F,20,D2,FF,C8,A9,AF,20,94,
04,C9,7F,0B29
890 DATAB0,1A,20,D2,FF,20,6E,17,
A5,AF,A4,B0,20,22,A2,20,6F,A4,20,
8B,90,A9,0A00
900 DATAFF,F0,03,4C,4C,17,29,7F,
20,D2,FF,A9,24,20,D2,FF,20,6E,17,
A9,22,20,098B
910 DATAD2,FF,A0,00,A9,AF,20,94,
04,AA,F0,1E,C8,A9,AF,20,94,04,85

```

```

,22,C8,A9,0B29
920 DATAAF,20,94,04,85,23,A0,00,
A9,22,20,94,04,20,D2,FF,C8,CA,D0,
F4,A9,22,0A44
930 DATA20,D2,FF,F0,51,D0,4F,90,
03,4C,7B,16,29,7F,20,D2,FF,C8,A9,
AF,20,94,0B2E
940 DATA04,C9,7F,B0,12,20,D2,FF,
20,6E,17,A9,46,20,D2,FF,A9,4E,20,
D2,FF,D0,0B3C
950 DATA29,29,7F,20,D2,FF,A9,25,
20,D2,FF,20,6E,17,A0,00,A9,AF,20,
94,04,85,095B
960 DATA62,C8,A9,AF,20,94,04,85,
63,A2,90,20,C9,A2,20,6F,A4,20,8B,
90,A9,0D,0A00
970 DATA20,D2,FF,1B,A5,AF,69,05,
85,AF,90,02,E6,B0,20,E4,FF,20,E1,
FF,D0,01,0BFB
980 DATA60,A5,C6,C9,40,D0,F1,3B,
B0,8F,A9,3D,20,D2,FF,1B,A5,AF,69,
02,85,AF,0BEE
990 DATA90,02,E6,B0,60,20,73,04,
20,85,94,A5,15,4B,A5,14,4B,20,E4,
9D,A0,01,0B9D
1000 DATAB1,14,AA,8B,B1,14,AB,6B,
85,14,6B,85,15,8A,4C,76,9A,A5,3,
B,8D,4B,1B,091A
1010 DATAA5,3C,8D,49,1B,20,73,04,
C9,29,F0,03,4C,AB,17,20,73,04,D,
0,01,60,A2,07C3
1020 DATA0E,4C,83,86,1B,9B,65,D0,
85,3B,A5,D1,69,00,85,3C,20,D2,1,
7,4C,33,1B,0B4B
1030 DATA20,73,04,20,4B,1B,8D,42,
1B,8C,43,1B,A5,0E,8D,44,1B,A5,0,
D,8D,45,1B,0620
1040 DATAA5,3B,A4,3C,8D,46,1B,8C,
47,1B,AD,4B,1B,AC,49,1B,85,3B,8,
4,3C,AD,42,0B1F
1050 DATA1B,85,49,AD,43,1B,85,4A,
20,73,04,20,2C,93,A5,3B,8D,4B,1,
B,A5,3C,8D,076E
1060 DATA49,1B,AD,44,1B,85,0E,4B,
AD,45,1B,85,0D,4B,A9,8E,4B,A9,9,
0,4B,AD,46,0B1C
1070 DATA1B,85,3B,AD,47,1B,85,3C,
60,20,79,04,C9,2C,F0,34,C9,29,F,
0,03,4C,8B,0B47
1080 DATA17,60,00,00,00,00,00,00,
00,00,00,A6,3B,D0,02,C6,3C,C6,3,
B,A2,00,24,04F3
1090 DATA4B,8A,BA,E0,2B,90,0E,20,
61,1B,60,A9,00,85,0D,20,73,04,4,
C,A5,96,4C,07D0
1100 DATA27,93,20,D2,17,4C,33,1B,
20,84,9D,E0,19,80,3A,86,D0,20,D,
B,9D,E0,2B,0971
1110 DATAB0,31,86,D1,20,D8,9D,E0,
2B,80,2B,86,D2,20,D8,9D,E0,19,8,
0,1F,86,D3,0B8B
1120 DATAA5,D0,C5,D3,80,17,A5,D1,
C5,D2,80,11,A5,D3,A6,D2,20,67,D,
E,A5,D0,A6,0E12
1130 DATAD1,20,7A,DE,4C,91,D8,4C,
1C,99,A5,14,4B,A5,15,4B,20,73,0,
4,20,85,94,0B02
1140 DATA20,67,9D,85,52,C9,10,F0,
07,C9,0B,F0,03,4C,A1,94,A9,00,8,
5,14,85,15,0BEC
1150 DATAA0,00,20,80,04,C8,C9,32,
B0,1C,C9,30,90,1B,6A,26,14,26,1,
5,C4,52,D0,0B69
1160 DATAEB,A5,14,A6,15,AB,6B,85,
14,6B,85,15,8A,4C,76,9A,4C,A1,9,
4,A5,14,4B,0972
1170 DATAA5,15,4B,20,73,04,20,8E

```

```

,94,20,14,93,20,E4,9D,20,8B,94,2,
0,24,19,6B,0747
1180 DATA85,15,6B,85,14,4C,CA,9C,
A9,10,20,5C,9B,A0,00,A9,00,06,1,
4,26,15,2A,06E5
1190 DATA69,30,91,62,C8,C0,10,D0,
F0,60,20,8B,1B,A5,14,4B,A5,15,4,
B,4C,10,85,094B
1200 DATA20,73,04,20,85,94,20,1B,
9E,A5,14,4B,A5,15,4B,20,E4,9D,2,
0,24,19,6B,0712
1210 DATA85,15,6B,85,14,4C,CA,9C,
20,73,04,20,4B,9C,C9,5B,90,03,4,
C,4C,CC,85,0B85
1220 DATAD0,A5,3B,4B,A5,3C,4B,A0,
00,20,80,04,99,00,02,C8,C4,D0,D,
0,F5,A9,00,09FA
1230 DATA99,00,02,85,3B,A9,02,85,
3C,20,53,89,20,14,93,6B,85,3C,6,
B,85,3B,60,073B
1240 DATA24,61,1B,10,23,A5,61,29,
7F,C9,02,80,1C,C9,01,1B,D0,16,A,
2,00,85,63,0797
1250 DATAC9,00,D0,0F,E8,E0,03,D0,
F5,A5,62,29,7F,C9,00,D0,02,3B,6,
0,4C,1C,99,0A1B
1260 DATA20,73,04,20,85,94,4C,17,
93,20,CC,19,20,A0,19,90,03,4C,0,
F,1A,A0,1A,0666
1270 DATAA2,3B,20,59,A2,20,27,A6,
A0,1A,A9,3B,20,5C,A0,A0,9F,A9,F,
0,20,66,A0,0997
1280 DATA20,E4,A5,A0,1A,A9,3B,20,
72,A0,4C,1A,AB,20,D5,19,A0,AA,A,
9,EC,4C,6C,0A2C
1290 DATAA0,A5,66,10,06,20,7F,1A,
4C,27,A6,4C,7F,1A,20,CC,19,A9,3,
9,A0,94,20,07B3
1300 DATA5C,A0,20,94,A2,A0,1A,A9,
33,20,21,A2,4C,97,A1,8B,34,00,0,
0,00,00,00,070B
1310 DATA00,00,00,20,CC,19,A0,1A,
A9,33,20,5C,A0,20,94,A2,A9,39,A,
0,94,20,21,0764
1320 DATAA2,4C,97,A1,20,E7,FF,A9,
00,20,8D,FF,A9,04,AA,A0,FF,20,B,
A,FF,20,85,0C25
1330 DATAA7,A2,04,20,97,A7,A9,0D,
20,D2,FF,20,79,04,20,FF,BA,A2,0,
4,20,C3,FF,0A20
1340 DATA4C,E7,FF,A9,EC,A0,AA,4C,
21,A2,20,84,9D,E0,02,90,03,4C,1,
C,99,86,E0,0B3D
1350 DATAA9,04,AA,A0,FF,20,BA,FF,
A9,00,20,8D,FF,20,C0,FF,A2,04,2,
0,C9,FF,A6,0C67
1360 DATAE0,F0,03,4C,8B,1C,AD,13,
FF,C9,D1,F0,0B,A9,11,8D,FE,07,2,
0,D2,FF,4C,0B03
1370 DATACB,1A,A9,91,8D,FE,07,20,
D2,FF,A9,00,85,22,A9,0C,85,23,A,
2,00,A0,00,098E
1380 DATA9B,85,4B,3B,B1,22,85,5E,
85,5D,3B,9B,4B,A5,4B,AB,A5,5E,2,
9,80,F0,09,098D
1390 DATA20,89,1B,A5,5E,29,7F,85,
5E,A5,5E,E9,20,1B,10,0B,69,60,9,
9,AC,1B,4C,0B03
1400 DATA2F,1B,69,20,3B,E9,40,1B,
10,0B,69,40,99,AC,1B,4C,2F,1B,6,
9,40,3B,E9,06CD
1410 DATA60,1B,10,0B,69,80,99,AC,
1B,4C,2F,1B,69,60,3B,E9,80,1B,1,
0,05,69,C0,072F
1420 DATA99,AC,1B,6B,AB,C8,E6,4B,
A5,5D,29,80,F0,03,20,9B,1B,C0,1,
4,D0,96,A0,0AB7

```



```

1430 DATA00,B9,AC,1B,20,D2,FF,C8
,C4,4B,D0,F5,E8,E0,32,D0,12,A9,9
1,20,D2,FF,0D14
1440 DATAA9,0D,20,D2,FF,A9,04,20
,C3,FF,4C,CC,FF,1B,AS,22,69,14,B
5,22,AS,23,0A1B
1450 DATA69,00,85,23,AS,22,29,04
,F0,03,4C,D2,1A,AS,0D,20,D2,FF,A
D,FE,07,20,0BA9
1460 DATAD2,FF,4C,D2,1A,BA,85,4C
,AS,4B,AA,AS,12,9D,AC,1B,E6,4B,A
5,4C,AA,CB,0BB1
1470 DATA60,BA,85,4C,AS,4B,AA,AS
,92,9D,AC,1B,E6,4B,AS,4C,AA,60,0
0,00,00,00,0920
1480 DATA00,00,00,00,00,00,00,00
,00,00,00,00,00,00,00,00,00,00
0,00,00,00,0000
1490 DATA00,00,00,00,00,00,00,00
,00,00,00,00,00,00,20,92,1C,AS,8
6,A0,1C,20,02D9
1500 DATA88,90,20,72,F1,AD,02,03
,8D,90,1C,AD,03,03,8D,91,1C,AS,0
E,20,A6,1C,080C
1510 DATA85,AC,8D,85,1C,AS,00,85
,AD,20,C0,FF,AC,85,1C,20,C6,FF,A
9,84,8D,2A,0B31
1520 DATA03,AS,1C,8D,2B,03,AS,1E
,8D,02,03,AS,1C,8D,03,03,20,CF,F
F,20,CF,FF,0810
1530 DATA20,CF,FF,85,14,20,CF,FF
,85,15,05,14,F0,36,AS,90,00,32,2
0,CF,FF,85,0AF8
1540 DATA14,20,CF,FF,85,15,A0,00
,20,CF,FF,99,00,02,AE,F6,07,E0,3
F,F0,19,C8,0A60
1550 DATAC9,AS,0D,00,EE,98,85,08,AS
,90,0D,0D,AS,02,A0,00,85,3C,84,3
B,A4,0B,4C,0987
1560 DATA36,87,AD,90,1C,8D,02,03
,AD,91,1C,8D,03,03,AS,08,8D,2A,0
3,AS,EF,8D,0825
1570 DATA28,03,AD,85,1C,20,C3,FF
,20,CC,FF,4C,03,87,60,00,4D,45,5
2,47,49,4E,0841
1580 DATA47,20,00,00,00,00,AS,E6
,20,21,CB,20,85,CC,AS,00,8D,78,0
2,85,0A,A0,0782
1590 DATA05,4C,3F,CA,A6,97,E0,00
,F0,0E,DD,0B,05,0D,06,38,E9,01,4
C,A6,1C,CA,092F
1600 DATAD0,F2,60,20,BF,C7,AS,0B
,20,D2,FF,AS,0D,20,D2,FF,AS,00,8
D,AF,1B,AS,08BA
1610 DATA06,8D,80,1B,AS,00,8D,AC
,1B,8D,AD,1B,AS,80,8D,B1,1B,AD,A
F,1B,8D,AE,09E4
1620 DATA1B,AD,AC,1B,8D,B2,1B,AD
,AD,1B,29,F8,8D,83,1B,AS,00,8D,B
4,1B,8D,B5,0A21
1630 DATA1B,AD,AE,1B,4A,4A,4A,AA
,F0,17,AS,00,AB,4B,1B,98,69,40,A
B,68,69,01,088C
1640 DATA4B,CA,00,F4,8C,85,1B,68
,8D,84,1B,AD,AE,1B,29,07,8D,86,1
B,AD,AD,1B,0A6F
1650 DATA29,07,8D,B7,1B,AS,07,38
,ED,B7,1B,8D,B7,1B,AS,00,6D,B
5,1B,AB,AS,08E4
1660 DATA20,6D,B4,1B,AA,98,6D,B3
,1B,AB,8A,6D,B2,1B,AA,98,6D,B6,1
B,8D,B9,1B,0A26
1670 DATA8A,69,00,8D,B8,1B,AC,B9
,1B,AD,B8,1B,85,D1,AS,00,85,D0,A
9,D0,20,94,0AD4
1680 DATA04,8D,BA,1B,AS,01,AE,B7
,1B,F0,04,0A,CA,D0,FC,2D,BA,1B,F

```

```

0,15,AD,AE,0A86
1690 DATA1B,3B,ED,AF,1B,8D,BB,1B
,AS,01,AE,BB,1B,F0,04,0A,CA,D0,F
C,1B,6D,B1,0A65
1700 DATA1B,8D,B1,1B,AD,AF,1B,1B
,6D,80,1B,EE,AE,1B,CD,AE,1B,30,0
3,4C,EB,1C,090E
1710 DATAAD,B1,1B,20,D2,FF,1B,AD
,AD,1B,69,01,8D,AD,1B,AD,AC,1B,6
9,00,8D,AC,09CC
1720 DATA1B,AS,01,CD,AC,1B,D0,07
,AS,40,CD,AD,1B,F0,03,4C,DA,1C,A
9,0D,20,D2,098B
1730 DATAFF,AD,AF,1B,1B,69,07,8D
,AF,1B,C9,C4,B0,03,4C,D2,1C,C9,C
B,F0,08,AS,0804
1740 DATA03,8D,80,1B,4C,D2,1C,AS
,0F,20,D2,FF,AS,0D,20,D2,FF,20,D
2,FF,AS,04,0A83
1750 DATA20,C3,FF,4C,CC,FF,20,DE
,B6,AS,3B,8D,33,1E,8D,4E,1E,AS,2
B,85,5F,AS,0ABD
1760 DATA2C,85,60,A0,01,84,0F,20
,6D,B1,F0,43,20,E4,FF,C9,03,F0,3
C,A0,03,A2,09C6
1770 DATAFF,84,03,EB,8D,02,02,F0
,33,C9,20,F0,F6,C9,22,F0,F2,C8,2
0,6D,B1,F0,0CB4
1780 DATA12,C9,20,F0,F6,C9,22,F0
,F2,5D,02,02,F0,DF,A4,03,CB,D0,D
6,A0,00,20,08B3
1790 DATA6D,B1,AA,CB,20,6D,B1,86
,5F,85,60,0D,B4,4C,CB,1E,A0,01,8
4,0F,20,3E,0983
1800 DATA90,CB,20,6D,B1,AA,CB,20
,6D,B1,84,49,20,5F,A4,AS,20,A4,4
9,29,7F,20,0954
1810 DATAB2,90,C9,22,D0,06,AS,0F
,49,FF,85,0F,CB,F0,D2,20,6D,B1,F
0,8D,10,E9,08D1
1820 DATAC9,FF,F0,E5,C9,FE,F0,3A
,D0,47,24,0F,30,0B,AA,84,49,A0,0
0,0A,F0,10,0C04
1830 DATACA,10,0C,E6,22,D0,02,E6
,23,B1,22,10,F6,30,F1,CB,B1,22,3
0,8B,20,82,0A1B
1840 DATA90,D0,F6,20,3E,90,A0,01
,AS,00,91,3B,8B,AS,3F,91,3B,A2,8
0,4C,83,86,0A0D
1850 DATAA2,11,86,23,A2,AF,86,22
,CB,20,6D,B1,4C,AE,1E,A2,81,86,2
3,A2,8E,86,098D
1860 DATA22,4C,AE,1E,20,84,9D,E0
,9F,90,03,4C,1C,99,86,D0,20,91,9
4,20,14,93,08E8
1870 DATA20,E4,9D,84,D1,85,D2,20
,42,1F,A2,0F,86,13,20,97,A7,1B,A
5,D0,69,60,09CC
1880 DATABD,50,1F,AS,D1,8D,51,1F
,AS,02,8D,52,1F,AS,D3,8D,53,1F,A
0,00,89,4F,0A03
1890 DATA1F,20,82,90,CB,C0,05,D0
,F5,4C,FE,90,AS,00,85,D3,20,9D,A
B,20,D8,9D,08AB
1900 DATAB6,D3,60,50,00,00,00,00
,20,6B,AB,AS,2B,A4,2C,85,D0,84,D
1,AS,2D,A4,08FC
1910 DATA2E,85,D2,84,D3,20,7E,1F
,20,D8,FF,AS,D0,A4,D1,85,2B,84,2
C,AS,D2,A4,08F5
1920 DATAD3,85,2D,84,2E,60,20,90
,1F,85,2C,84,2B,20,90,1F,4B,9B,A
A,6B,AB,AS,08DB
1930 DATA2B,60,20,9D,AB,20,91,94
,20,14,93,4C,E4,9D,EA,EA,EA,20,4
B,9C,85,D0,0AEO
1940 DATAB6,D1,84,D2,20,F7,1F,A0

```

```

,00,C4,97,F0,0B,89,1D,05,C9,6F,F
0,1F,CB,4C,080F
1950 DATAAD,1F,AS,0F,20,A6,1C,85
,D4,A0,0F,AS,D3,20,BA,FF,AS,00,2
0,8D,FF,20,0A65
1960 DATAB5,A7,A6,D4,4C,E1,1F,89
,13,05,C5,D3,D0,DA,8E,09,05,20,9
7,A7,AS,D0,08A4
1970 DATAA6,D1,86,22,A4,D2,84,23
,20,8E,90,20,3E,90,4C,FE,90,20,7
9,04,D0,05,09B4
1980 DATAA2,0B,86,D3,60,20,D8,9D
,E0,0B,80,F6,4C,1C,99,0C,06,25,0
9,06,02,04,07D3
1990 DATA12,0B,09,0B,12,15,2C,06
,0B,12,1E,2F,47,7A,7E,BF,DC,E0,E
9,F3,FD,09,0790
2000 DATA13,1B,34,5C,61,5C,5E,60
,62,64,66,6B,6A,6C,6E,70,72,74,7
6,7C,7E,80,0857
2010 DATAB2,84,86,8B,8A,8C,8E,90
,92,94,96,9B,9A,9C,9E,A0,A2,A4,A
6,FF,0B,15,0C1B
2020 DATA19,77,8E,9B,EB,F6,FD,4E
,54,71,74,EE,F1,3A,71,6B,AS,B9,C
6,01,16,2F,0877
2030 DATAAS,AA,B4,CE,D1,D7,DA,DD
,E2,E7,EE,F1,F4,F7,FE,03,10,15,1
B,1E,2A,2F,0D7B
2040 DATA40,5F,70,73,1A,3E,5C,D7
,DA,DF,E1,EB,FC,07,15,1B,1E,2A,3
F,41,AD,BF,09F9
2050 DATAEC,FE,01,0E,11,1E,21,2E
,3C,45,7A,8B,93,AS,D6,DA,E6,EC,F
1,F6,00,0A,0AA5
2060 DATA14,64,6A,7A,86,CC,D1,D6
,D9,DE,E1,E4,E7,EA,ED,F2,F7,FA,F
D,16,1A,1D,0EEC
2070 DATA22,25,2A,30,33,39,3F,44
,49,4E,51,57,5A,5D,6B,70,79,7E,8
2,85,8A,94,077D
2080 DATA97,9A,9E,A1,A4,AS,AC,B3
,BB,BB,C0,C5,CC,D1,D9,DF,E6,EF,F
2,0F,12,69,0EBA
2090 DATADF,EC,F7,13,22,27,2C,31
,36,69,AA,8B,C0,D6,20,8B,90,A0,0
0,89,0D,21,09D4
2100 DATA99,00,06,CB,C0,8D,D0,F5
,60,78,8D,3F,FF,85,D2,C9,10,D0,1
5,AS,1B,8D,087F
2110 DATA46,06,AS,69,8D,47,06,AS
,20,8D,35,06,8D,3C,06,4C,31,06,A
9,3B,8D,46,06CF
2120 DATA06,AS,E9,8D,47,06,AS,50
,8D,35,06,8D,3C,06,A2,00,8D,0B,2
0,85,D0,A0,08B1
2130 DATA00,B9,1B,20,85,D1,CB,84
,D3,A0,00,B1,D1,1B,69,30,91,D1,A
4,D3,C4,D0,08A9
2140 DATAD0,E9,1B,9B,6D,3B,06,8D
,3B,06,90,03,EE,3C,06,E8,E6,D2,E
0,10,D0,CE,0AD6
2150 DATAAS,D2,C9,40,80,03,A2,40
,2C,A2,10,8E,0D,03,8E,0F,03,8E,1
1,03,8E,09,076A
2160 DATA03,8E,03,03,8E,6B,13,EB
,8E,0B,03,AS,1B,8D,3B,06,8D,3E,F
F,5B,60,00,073B

```

PROGRAM: +4 EX BASIC 2

10 REM LISTING2  
20 SCNCLR



29



# Word Processor Roundup

*As a word processor is defined as a simple means of entering, editing, printing, and storing text on a computer why are there so many of them? This article covers no less than twelve different word processors, each offering a variety of functions and options to cater for the incredible variety of uses these essential programmes are used for.*

**I**f you have ever written a letter, memo, note or magazine article then you need a word processor and so you will no longer be buried in 'early drafts' and pools of correcting fluid. With a word processor you can type on the keyboard anything you would on a typewriter and then correct it, alter it, save it for later use and print it out.

In this article (which I wrote using PaperClip) I have looked at twelve word processors and assembled their strengths

and weaknesses and included certain factors and features that may help you find the word processor that is best suited for your needs. (Table 2).

## EASY SCRIPT/Commodore

Easy Script is probably one of the best known C64 word processors as it was

given away free with the 1541 disk drive.

Although Easy Script contains facilities to create and edit documents, it uses a system of control commands that means that text on the screen isn't the same as it will appear on the paper. In fact the program includes a special function to show the document as it will actually appear. Since then the word processors have become friendlier which means you can get started without wading through a huge manual.

### • Ski Writer

```

Line 10      Editor      Column 1
COMMENT= This resume uses the "I"
characters for temporary paragraph
indents. Select PREVIEW to see the
effect.
JOSEPH BROWN      55 Dee Road
                  S. Boston, MA 02127
                  (617) 555-1234
Experienced
1983 - present    Public Beach Group
                  South Boston, MA
at Carson Beach, Full-time Lifeguard
months of June and September.
1981 - 1982      Jimmy's Variety
                  South Boston, MA
F1 Help          F3 Edit Menu      STOP Main Menu
  
```

### • Master Word

```

Dear (John/Jane):
< 6
I don't write very often so I'm not
real good with written < 7
words. I am not angry with you for any
thing, nor am I unhappy with < 8
the way our relationship has been going.
I must, however say that < 9
I'm just not ready for a full-time rela-
tionship with anyone yet. < 10
- < 11
I have an entire life ahead of me,
and right now I feel like I < 12
have to move around and meet new people
before I settle down. < 13
I cannot say that I don't love you, be-
cause I do, very, very much. < 14
Someday I will probably ask you to mar-
ry me but for now, it's a < 15
scary thing to wonder if I'm doing the
right thing by loving myself. < 16
F2=Help F1=Menu Line# 12
  
```



## SKI WRITER/Mastertronic

Mastertronic's imported Ski Writer includes only the briefest of manuals (three pages of a minute leaflet) and consists of four main functions that are assessed through a main menu.

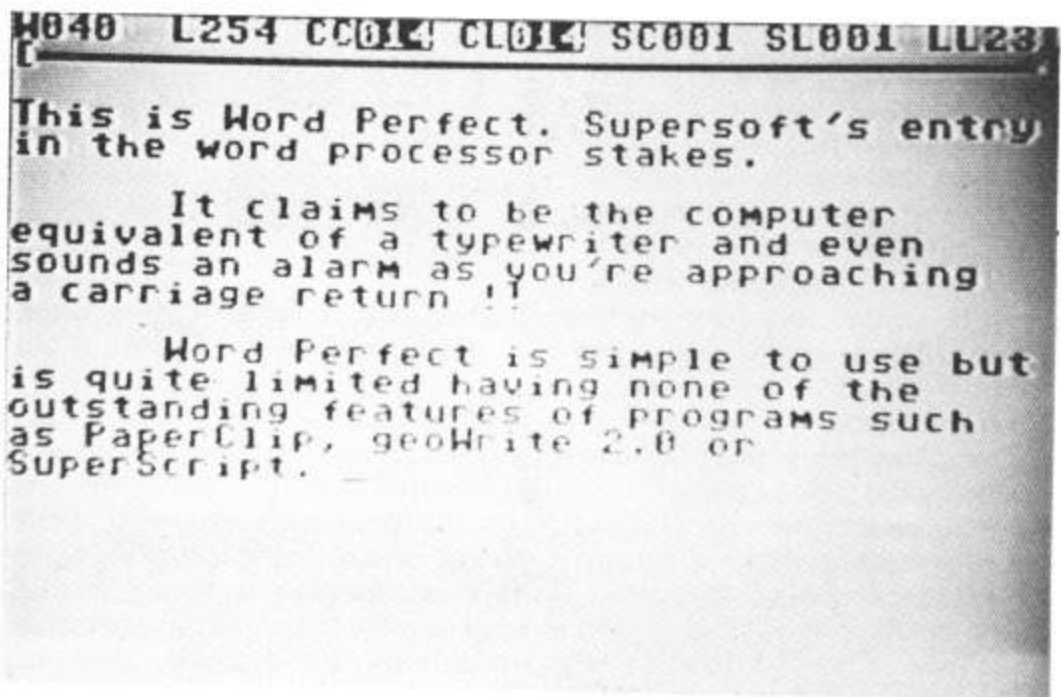
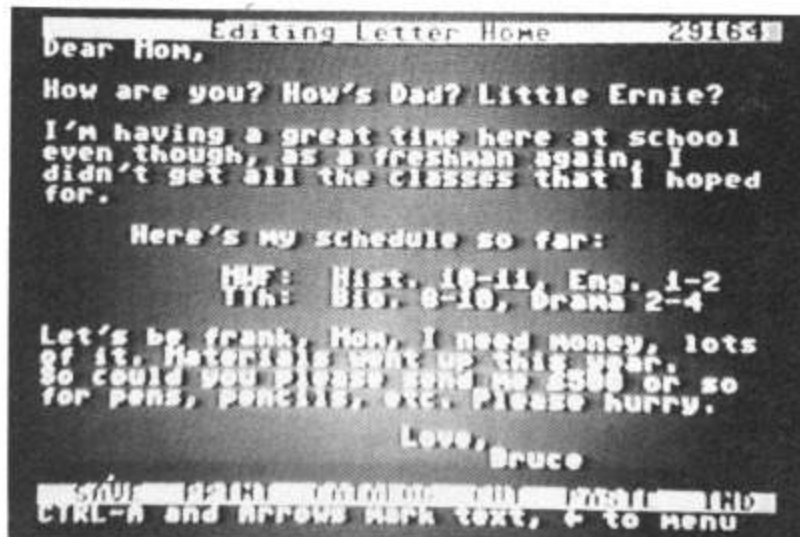
The Edit option allows you to type and edit your document and format it, using Easy Script style commands to set margins, page numbers, headers and text justification. The Preview mode interprets these commands and displays the text as it would appear on paper. Your document can then be saved, loaded or merged from disks with the file option that can also format blank disks before printing on any defined printer and paper type.

## MASTER WORD/Argus Press Software

As part of the Load 'N Go series of cheap and cheerful disk programs Master Word has the distinction of being the only word processor that is supplied with no instructions at all!

If this complete lack of documentation hasn't put you off you'll find pages and pages of help screens to get you going as well as files of sample letters which include nine different business letters, nine letters home, five true love notes, two, 'the romance is' over and one, 'it never really began' letter for the one you hate!

### • Cut & Paste



### • Word Perfect

If you don't expect too much from Master Word you won't be disappointed.

## WORD PERFECT/Supersoft

Word Perfect is an easy to use program but has its limitations including a 254 line limit to documents. This is just over four A4 pages and so limits its applications to short notes, letters and memos. Longer articles such as this one just wouldn't be possible.

If you stay within this limitation you'll be able to enter and edit text using simple commands that try to mimic a typewriter. You even hear a bell when

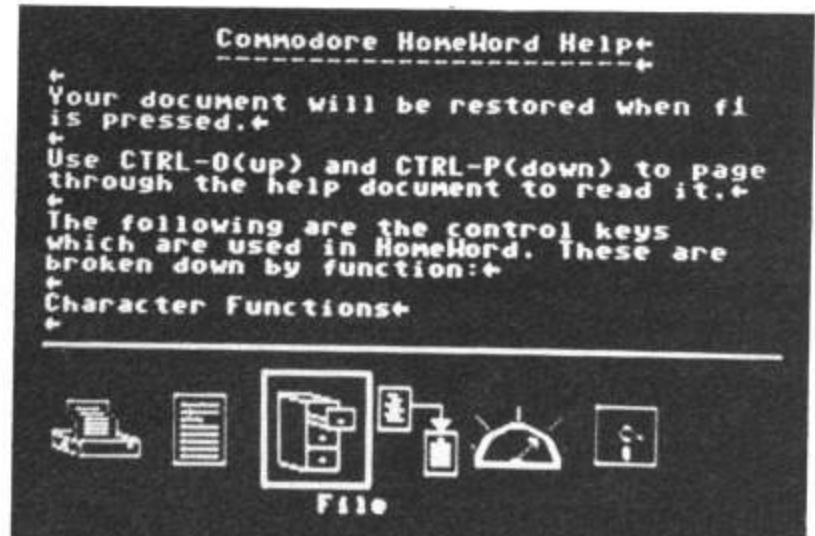
you're hearing a carriage return although you can let the program automatically wordwrap to a new line.

## CUT & PASTE/Ariolasoft (Electronic Arts)

Many word processors claim to be easy to use but this one actually is so much so, that you can load in a sample document, edit it, add in your own words of wisdom and save it without looking at the slim introduction manual!

Some word processors include a command bar; Cut & Paste has two! At

### • Homeword





the top of the screen you get a menu of files on disk and at the bottom a duck-shoot command bar to load and save these files and format blank disks.

Once you're actually editing text, whether it's one of the supplied formatted letters (to Mom asking for money!), resume or memos or all your own work, then cut and paste is the name of the game. Any word, sentence or block of text can be defined and cut from the document and then pasted in another part of it.

This means you can move and copy text as well as restore deleted words that you really wanted (as long as you haven't cut anything else, or even cut a general introduction that can then be pasted onto a whole series of different documents.

## HOMESWORD/Green valley

Homeword (also sold as Master Writer) is the only icon controlled word processor in this collection. It has a picture of a filing cabinet for its load, merge and save options as well as a printer, a page of text for move, copy, erase and find and replace edit options, a layout icon to set text justification and a disk to catalog and create files.

Selecting any icon leads to another set of icons and so on until you find the option you need. As with Cut and Paste, the manual is almost redundant as it's always clear through icons and screen prompts, what you're doing, how you got there and how to get out of it without losing all your work.

To help you further a display, at the bottom of the text entry screen shows the current page, a bar illustrates the remaining memory and a bit pattern representation of your page is shown.

## TEAM-MATE/Tri Micro

Team-Mate is in fact the C64 version of the programmes supplied with Plus/4 and features an integrated package of a word processor, spreadsheet, database and graphics package (that produces bar and pie charts).

The word processing part of the package is undoubtedly limited with each file restricted to only 99 lines. However, more than one file can be linked together to form longer documents that can use the output from graphics and spreadsheet packages in its documents. It doesn't compare with the full word processors but is a useful addition to a package.

## TRI WORD/

Tri Word is part of the Triangle (word processor/spreadsheet and file handler package) and is less limited than its Team-Mate competitor with up to 400 lines in a file. However data cannot be mixed between the programmes, and files can't be linked together. However you shouldn't need to as the files should be big enough. One thing that remains a mystery is why F1 moves the page down and F7 moves it up?

## MINI OFFICE II/Database

The Mini Office II word processor is probably the best word processor supplied in an integrated package.

From a single menu you can edit and create your text, preview it, search and replace consistent spelling errors and load and save your work. Meanwhile a bar at the top of the screen keeps track of the various modes you are in, the time it has taken you to create the document and an extremely useful word count. In fact everything you'd expect from an ordinary word processor which you get with a spreadsheet, database, graphics package, label printer and communications package!

## GEOWRITE 2.0/First Analytical

GeoWrite 2.0 is just part of the Writer's Workshop package that runs with the pull-down menus and pointers of the Mac like GEOS operating system.

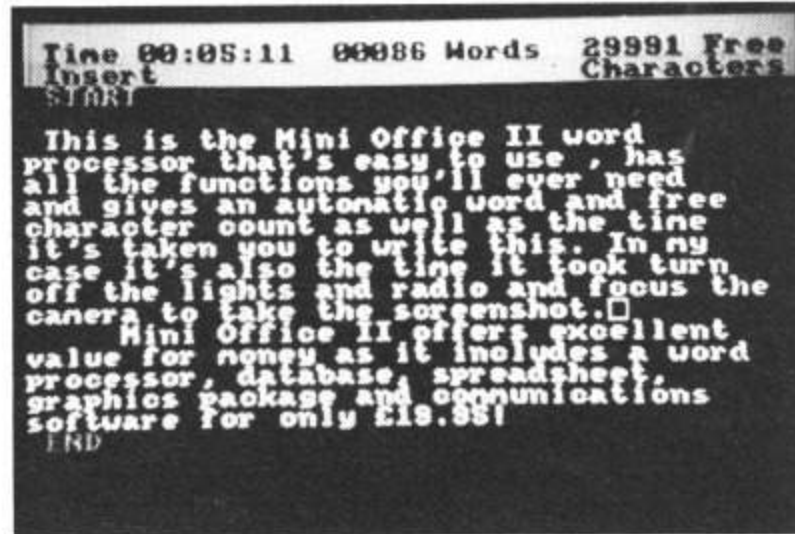
A version of geoWrite was included with the GEOS master disk but this was little more than a text handler and has been replaced by a full word processor featuring headers and footers, left, right and centre justification, plain, bold, italic, outline, subscript and superscript text as well as a choice of fonts in a variety of point sizes and the incredibly powerful text grabber.

The text grabber can convert any C64 word processor document into

### • Tri Word



### • Mini Office II







● Geowrite

geoWrite format which can then be edited by geoWrite and improved by adding different fonts and geoPaint graphics.

### PAPERCLIP/Ariolasoft (Batteries Included)

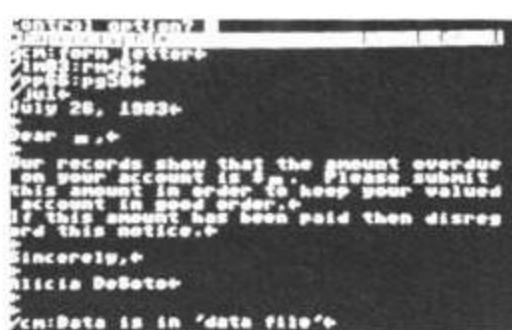
I used Paperclip to write this article because I found it the easiest machine to use. I needed a word processor that didn't have any complicated control codes, that you could accept, type directly onto the screen and could easily correct mistakes as you go. Also which had a spell checker to remove any spelling mistakes before being saved to disk, for direct entry into a typesetting machine.

I didn't want to be restricted to a small number of lines but I needed to know how many words I had used. (The Spellchecker is only available in the special Paperclip with Spell Checker pack which is definitely worth the extra fiver.)

The spell checker is the secret of Paperclip's success as it compares these words with its 15,000 word dictionary and prompts you to alter any it can't find or understand. Therefore it can spot typing errors, spelling mistakes and words run together through missed spaces, while giving you a report of the total number of words used and their average length.

This time I didn't need any of its other features such as transferring text between different documents, sorting capabilities, editing data from databases and spreadsheets or the ability to create form letters. But I did use the special commands to construct the comparison table. (Table 2).

Paperclip with the added spell checker is more expensive than the others but you do get more for your money. Great value and a must for journalists with deadlines!



● Paperclip

### SUPERSCRIPT/Precision

I may prefer Paperclip but there are others such as Your Commodore's illustrious Editor who prefers Superscript to create his words of wisdom.

Superscript is undoubtedly an incredibly powerful word processor that's equal to Paperclip in the feature it offers including a spell-checker and mail-merge facility.

If there's any difference then it's how the commands are accessed. In Paperclip these are through conversational two key commands, whereas Superscript employs a series of nested duckshoot menus. For example, selecting Document from the



● Superscript

main menu leads to a sub-menu that includes options to load, save, directory and spell. Select spell and you're led to another menu and options to check spelling, view or print the user dictionary and display a word count.

Comparing Superscript and Paperclip is like the comparison between a Ferrari and a Porsche. They are both equally impressive but different to drive.

The following table compares the tested word processors with eight factors that are important to a buyer. The final two factors are an opinion of the programmes themselves and their manuals, and take into account, friendliness and general ease of use.

	Easy Script	Ski Writer	Master Word	Word Perfect	Cut & Paste	Home-Word
Headers	Yes	Yes	No	No	Yes	Yes
Footers	Yes	No	Yes	No	No	Yes
Line Spacing	Yes	Yes	Yes	No	Yes	Yes
Search/Replace	Yes	No	No	No	No	Yes
Help Function	No	No	Yes	No	No	No
Spell Checker	No	No	No	No	No	No
Word Count	No	No	No	No	No	No
Price		£4.99		£19.94		
Program	Fair	Fair	Good	Fair	Good	Good
Manual	Poor	Poor	None	Fair	Good	Good

	Team Mate	Tri Word	Mini Office	Geo-Write 2	Paper Clip	Super Script
Headers	No	Yes	Yes	Yes	Yes	Yes
Footers	No	No	Yes	Yes	Yes	Yes
Line Spacing	Yes	No	Yes	Yes	Yes	Yes
Search/Replace	Yes	Yes	No	Yes	Yes	Yes
Help Function	No	Yes	No	Yes	No	Yes
Spell Checker	No	No	No	No	No	Yes
Word Count	No	No	Yes	No	Yes	Yes
Price			£19.95	£37.50	£44.95	£39.95
Program	Fair	Fair	Good	Good	Great	Great
Manual	Fair	Fair	Good	Good	Good	Good



# Everyman's Guide to Graphics

*Graphics are a fascinating application for the C64. In this comprehensive guide, we point the way to better understanding and use of this facility.*

*By Allen Webb*

In my view, the crucial part of any piece of software is the graphics. There are very few items which need no attention to graphics, with even a text only package being improved by a redesigned font.

In this article, I want to give a detailed run down of the C64's graphics capability and how you can use it. Where it simplifies life, I will give listings of helpful routines

## Vic Chip

First, let us consider the driving force behind graphics, the VIC-II chip. This chip controls the graphics system which can in turn be altered via a number of registers. These registers are memory mapped allowing you to change them easily. Table 1 lists the most useful registers.

That's a pretty meaty lump of information and it's only provided as reference material. The rest of this piece will show you how the more important registers are used.

If you want to use your 64 efficiently, an appreciation of how it handles its memory is necessary. Figure 1 gives a simple memory map. The memory map can be considered to consist of two layers. The bottom layer is a block of 64K of RAM. On top of

this are superimposed two areas of ROM and other chips. Since different

devices occupy the same addresses, a register at address one is used to decide

Table 1

Register	Bit	Function
\$D000-\$D010 (53248-53264)		Sprite positions
\$D011 (53265)	7	Raster Compare
	6	Extended colour mode
	5	Bit map mode
	4	Blank screen
	3	24/25 row text
	2-0	smooth scroll Y direction
\$D012 (53266)		Raster Read/write
\$D013-\$D014 (53267-53268)		Light Pen registers
\$D016 (53270)	4	Multicolour mode
	3	38/40 column text
	2-0	Smooth scroll X direction
\$D017 (53271)		Y expand register
\$D018 (53272)		Memory Control Register
	7-4	screen matrix
	3-1	Character table
\$D019 (53273)		Interrupt register
\$D01A (53274)		IRQ mask register
\$D01B (53275)		Sprite priority register
\$D01C (53276)		Sprite colour mode register
\$D01D (53277)		X expand register
\$D01E (53278)		Sprite to sprite collision register
\$D01F (53279)		Sprite to background collision register
\$D020 (53280)		Screen border colour
\$D021-\$D024 (53281-53284)		Background colour registers
\$D025-\$D026 (53285-53286)		Sprite multicolour registers
\$D027-\$D02E (53287-53294)		Sprite colour registers



which are switched in. In normal use, the RAM under the ROMs is unavailable to Basic but it can be used for graphics.

The 64 treats the block of RAM as four banks of 16k:

Bank0-\$0000-\$3FFF(0 to 16383).

Bank1-\$4000-\$7FFF(16384 to 32767)

Bank2-\$8000-\$BFFF(32768 to 49151)

Bank3-\$C000-\$FFFF(49152 to 65535)

Bank 0 is the default bank. The bank in use is specified in bits 0 and 1 of location \$DD00.

The VIC can only address one bank at a time and it expects to find an area of screen memory and a character set within the bank. This approach offers almost unlimited flexibility but also makes the use of graphics in the default bank restricted.

Since the CPU and the VIC chip operate independently, the CPU doesn't care which bank is used for graphics. We can therefore reconfigure the machine from Basic very easily.

Let us consider how to reconfigure the memory map.

## Changing the BANK

This is achieved easily by changing the register at \$DD00:

```
10 POKE 56578, PEEK(56578) OR 3
20 POKE 56576, (PEEK(56576) AND 252) OR (3-BN)
```

Line 10 prepares the ground and line 20 switches in BANK number BN.

The VIC chip ignores the absolute address of the bank and uses only the relative addresses within the bank, i.e. each bank ranges from \$0000 to \$3FFF.

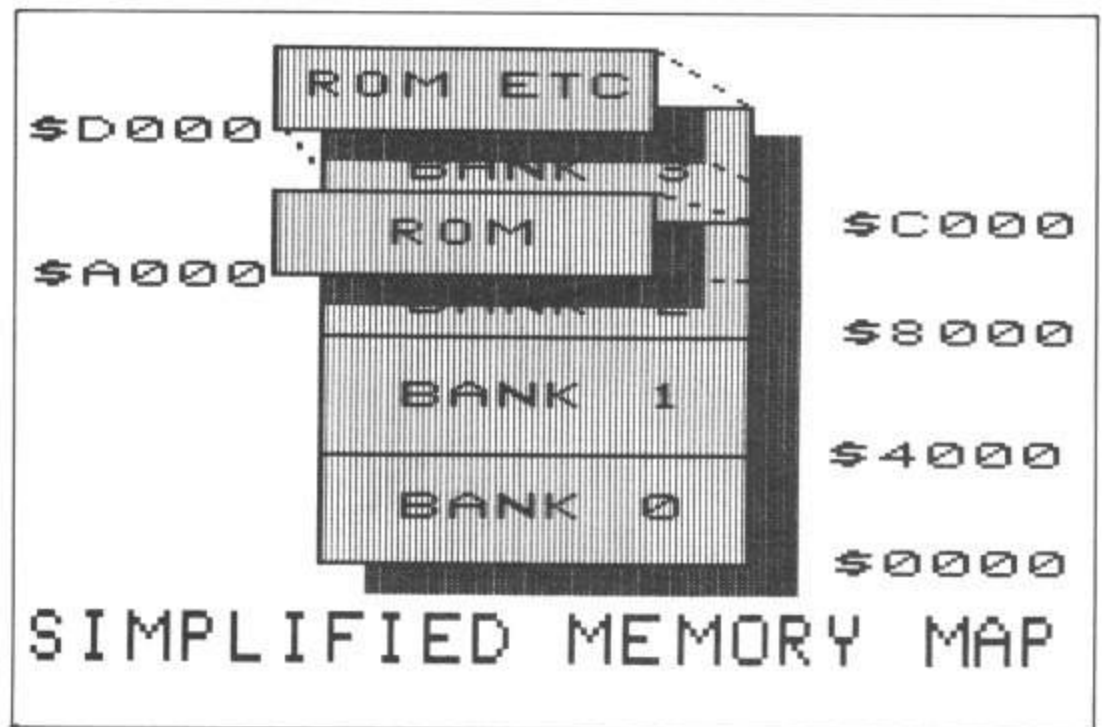
## Moving the Character Set

The register at 53272 tells the VIC chip where to get its character data. In fact, bits one to three hold this information.

This information is changed by:

```
POKE 53272, (PEEK(53272) AND 240) OR X
```

X is equal to the start address of the character data divided by 1024. With



only three bits used, only eight character sets are possible i.e.  $x = 0, 2, 4, 6, 8, 10, 12, 14$ .

Since the machine powers up with a character set, there must be default information somewhere. In fact, the default character set is held in ROM. This data is imaged to banks 0 and 2 and is found at the following addresses:

\$1000-\$17FF (Lower case set  $X=4$ )  
\$1800-\$1FFF (Upper case set  $X=6$ )

Clearly, it is possible to have a number of different sets of characters in a bank and simply switch between them as needed.

## Moving the Screen

The screen comprises of 1000 bytes of contiguous memory which usually resides between locations 1024 and 2024. This position is specified in bytes 4 to 7 in location 53272. These bytes actually specify the position of the screen in any bank of memory, and can be changed by:

```
POKE 53272, (PEEK(53272) AND 15) OR Y
```

Y is equal to the start address of the screen divided by 64. This time we have four bits in the register allowing 16 possible screen positions with Y

ranging from 0 to 240 in increments of 16. Unfortunately, you cannot use all RAM areas for the screen. If you use the areas imaged by the character ROM, you will get garbage on the screen.

In addition to changing the VIC register, you must also tell the operating system where the screen is. This is done with:

```
POKE 648, SCREEN/256
```

where SCREEN is the start address of the screen area.

The screen colour matrix cannot be moved and, in fact, presents no difficulties.

Listing 1 allows you to reconfigure your 64. The first part asks you to specify where the screen and character set are to go. These values are checked to ensure that they are in the same bank and are not at the same address. It doesn't check any further so beware. Line 60 to 80 calculate the register values. Line 90 checks to see if you need to copy down the character set and lines 100 to 150 do this job if required. Lines 160 to 190 reconfigure the machine.

### Listing 1: Reconfigure

```
26 10 PRINT CHR$(147); INPUT "SCREEN POSITION"; SCREEN
44 20 INPUT "CHARACTER SET ADDRESS"; CHARS
```

**Table 2**

Pixel one	Pixel two	Colour Register
clear	clear	53281
clear	set	53282
set	clear	53283
set	set	colour matrix

this mode, any given cell may contain only two colours; the background or paper colour and the foreground or ink colour. Any set pixel will have the ink colour and any unset pixel will have the paper colour.

In character mode, the paper colour is held in VIC register 53281 and the ink colour is held in the colour matrix.

This mode allows the greatest detail, albeit at the most limited colour range.

## 2. Multi-colour Mode

In this mode, pairs of pixels are used to define dots of colour. Since there are four possible arrangements for two pixels, four colours are allowed in any given character cell (Table 2).

This mode is a lot coarser but allows greater colour flexibility.

## Extended Background

This mode uses high resolution but offers four different paper colours in addition to the usual ink colours. The paper colour is determined by the POKE value of the character used and limits you to 64 different characters (Table 3).

## Redefined characters

OK, we've done the spade work, let's now look at the use of user defined characters.

You will have realised that the shape of characters is held in a table of data. Exactly how is of essence. Consider Figure 2. This shows a character design. The design comprises of eight lines of dots. Imagine that each set pixel is a 1 and each clear pixel is 0. That being so, the top line becomes 00111100. The decimal equivalent of this binary number is 60. Similarly, each line can be converted to a data value. The character table comprises of a sequence of data values for each character. The first eight data values in

```

FO 30 IF SCREEN = CHARS THEN ES
    = "ERROR..CHARS AND SCREEN AT
    SAME ADDRESS":GOTO60000
A1 40 IF INT(SCREEN/16384)=INT(
    CHARS/16384)THEN60
34 50 ES="ERROR..CHARS AND SCRE
    EN NOT IN SAME BANK":GOTO600
    00
1B 60 X=INT(SCREEN/16384)
5B 70 Y=(SCREEN-X*16384)/64
9B 80 Z=(CHARS-X*16384)/1024
E1 90 IF Z<>4 AND Z<>6 THEN
8E 100 PRINTCHR$(147)"COPYING R
    OM CHARACTER SET...THIS WILL
    [SPC3]TAKE A WHILE"
F4 110 POKE56334,PEEK(56334)AND
    254
4F 120 POKE 1,PEEK(1) AND 251
A2 130 FOR I=0TO2047:POKECHARS+I
    ,PEEK(53248+I):NEXT
F6 140 POKE1,PEEK(1)OR4
03 150 POKE56334,PEEK(56334)OR1

17 160 POKE56578,PEEK(56578)OR3

12 170 POKE56576,(PEEK(56576)AN
    D252)OR(3-X)
3B 180 POKE53272,Y+X
B3 190 POKE648,SCREEN/256
C3 200 PRINTCHR$(147)"DONE"
52 210 END
FD 60000 PRINT:PRINTES
9C 60010 GET I$:IF I$=""THEN6001
    0
3B 60020 GOTO10
  
```

In my view, the crucial part of any piece.

Run Listing 1 putting the screen at 50176 and the character table at 51200 and then enter the line:

POKE 44,4: POKE 1024, 0: NEW

You will have a machine offering 39933 bytes for Basic and 144 sprites. That's a lot more than you get on switch on! This extra capacity is achieved by:

- 1) Using BANK 3 and moving the screen and character set to a handy block of RAM between the ROMs.
- 2) Moving the start of Basic program

### Listing 2

```

10 DATA 60, 34, 34, 60, 34, 34, 60, 0
20 CH=4: FOR I=0TO7: READ X
30 POKE 51200+8*CH+1,X
40 NEXT
  
```

store down to 1025. Since we've moved the screen we can use the normal screen area for Basic.

3) You can use the memory behind the Kernal ROM (\$E000 to \$FFFF) and the remaining memory between the ROMs (\$C000 to \$C3FF) for sprites.

Machine code users don't have such a tough time since they aren't constrained by where they have to put their programs. It is, nevertheless, useful to reconfigure the machine.

## Graphics Modes

Before we launch forth into graphics handling, we must consider the graphics modes available to us. The screen occupies 1000 bytes and is divided into 64000 addressable points or pixels. There are two graphics modes allowing manipulation of the screen.

### 1. Character Mode

In this default mode, the screen uses 1000 characters, each occupying an 8x8 pixel cell.

### 2. Bit mapped Mode

In this mode, the screen uses a 320 by 200 array of pixels. Using this mode it is possible to create pictures and other images.

The fundamental difference between these modes is that character mode is supported by the operating system whereas bit map mode has no software to drive it. Both modes use 8x8 cells to control the colours used.

In addition to the graphics modes, there are three colour modes.

### 1. High resolution Mode

This is the default graphics mode. In

### POKE CODE

0-63  
64-127  
128-191  
192-255

### COLOUR REGISTER

53281  
53282  
53283  
53284



the table is used by the character normally used by @. The second block of eight is used by the character A. And so on. For any given character CH, its data values start at:

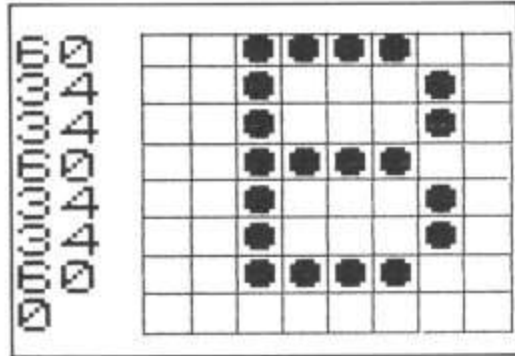
TABLE ADDRESS + CH\*8

As an experiment, run Listing 1 as before putting the character table at 51200. Then type in and run Listing 2.

Note what happens to the letter D.

Using this approach is rather slow. Listing 3 gives a machine code alternative.

This code lives at \$SC000 allowing you to use a relocated screen and character set as earlier. The code has



two routines. The first copies the ROM characters to a specified address, rather like lines 110 to 150 but faster. You call the routine with the command:

SYS 49152, ADDRESS

## Listing 3

```

77 2000 FOR L=0 TO 15: CX=0: FORD=OT
    015: READA: CX=CX+A: POKE 49152+
    L*16+D, A: NEXT D
82 2010 READA: IF A<>CX THEN PRINT "
    ERROR IN LINE": 2040+(L*10): S
    TOP
OF 2020 NEXT L: END
0D 2040 DATA 76, 6, 192, 76, 153, 192
    , 32, 196, 192, 169, 216, 141, 131,
    3, 169, 208, 2152
C4 2050 DATA 141, 130, 3, 169, 0, 133
    , 253, 173, 130, 3, 133, 254, 165, 2
    0, 133, 251, 2091
67 2060 DATA 165, 21, 133, 252, 141,
    232, 3, 141, 233, 3, 173, 14, 220, 4
    1, 254, 141, 2167
2C 2070 DATA 14, 220, 165, 1, 41, 251
    , 133, 1, 160, 0, 177, 253, 145, 251
    , 165, 251, 2228
A0 2080 DATA 24, 105, 1, 133, 251, 16
    5, 252, 105, 0, 133, 252, 24, 165, 2
    53, 105, 1, 1969
85 2090 DATA 133, 253, 165, 254, 105

```

```

, 0, 133, 254, 165, 253, 208, 222, 1
65, 254, 205, 131, 2900
4B 2100 DATA 3, 208, 215, 165, 1, 9, 4
    , 133, 1, 102, 21, 173, 14, 220, 9, 1
    , 1279
0B 2110 DATA 141, 14, 220, 173, 0, 22
    1, 41, 3, 168, 173, 232, 3, 56, 249,
    149, 192, 2035
1C 2120 DATA 141, 232, 3, 78, 232, 3,
    78, 232, 3, 173, 24, 208, 41, 240, 1
    3, 232, 1933
C7 2130 DATA 3, 141, 24, 208, 96, 192
    , 128, 64, 0, 32, 227, 192, 32, 206,
    192, 169, 1906
D0 2140 DATA 0, 141, 237, 3, 32, 253,
    174, 32, 138, 173, 32, 247, 183, 17
    2, 237, 3, 2057
1F 2150 DATA 165, 20, 145, 251, 200,
    238, 237, 3, 192, 8, 208, 232, 96, 1
    73, 238, 3, 2409
BE 2160 DATA 141, 239, 3, 96, 32, 253
    , 174, 32, 138, 173, 32, 247, 183, 9
    6, 6, 251, 2096
4F 2170 DATA 38, 252, 6, 251, 38, 252
    , 6, 251, 38, 252, 24, 165, 252, 109
    , 233, 3, 2170
E4 2180 DATA 133, 252, 96, 32, 196, 1
    92, 165, 20, 133, 251, 169, 0, 133,
    252, 96, 32, 2152
34 2190 DATA 144, 97, 32, 134, 97, 16
    5, 20, 133, 251, 173, 136, 2, 24, 10
    5, 4, 133, 1650

```

Where ADDRESS is the start of the character table. The routine also remembers where the character table is so always call it first in your program before trying to use the second routine.

The second routine redesigns a specified character and has the form:

SYS 49155, CH, B1, B2, B3, B4, B5, B6, B7, B8

Where CH is the character number and B1 to B8 are the bytes defining the character. To redefine D as B as done earlier, the command is:

SYS 49155, 4, 60, 34, 34, 60, 34, 34, 60, 0

For a bit of a laugh try this one:

```

10 SYS 49152, 51200
20 FOR I=0 TO 999: POKE
50176+I, 0: NEXT
30 FOR I=0 TO 7: A(I)=RND(I)*
256: NEXT
40 SYS 49155, 0, A(0), A(1), A(2), A(3),
A(4), A(5), A(6), A(7)
50 GOTO 20

```

Listing 3 works equally well in multicolour and extended background modes. To turn on multicolour character mode you must use:

POKE 53270, PEEK(53270) OR 16

To turn it off, use:

POKE 53270, PEEK(53270) AND 239

To turn on extended background mode, you must use:

POKE 53265, PEEK(53265) OR 64

To turn it off, use:

POKE 53265, PEEK(53265) AND 191

## Bit Mapped Mode

This mode is a bit of a paradox. Whilst on one hand it offers the greatest scope for artistic creativity, it is also memory hungry. In essence, it uses two or three blocks of memory.

### 1. The bit map

This is an area of 64000 pixels arranged in 200 lines of 320. This occupies 8K of RAM.

### 2. The colour array

This holds the colour information and comprises of 1000 character cells.

### 3. The color matrix

Multicolour mode uses this area to hold one of the colours.

The concepts discussed earlier with respect to telling the VIC chip where the bit map lies also apply here. Register 53272 holds the details of the bit map area (bits 1 to 3) and the colour array (bits 4 to 7). In other words the bit map occupies the character set area and the colour array occupies the screen memory.

To turn on the bit map, you must turn on bit 5 of register 53265:

POKE 53265, PEEK(53265) OR 32

Register 53270 decides whether multicolour or high resolution modes is used:

Multicolour on: POKE 53270, PEEK(53270) OR 16

Multicolour off: POKE 53270, PEEK(53270) AND 239

So how do we select the colours? In high resolution mode, the colour array

holds this information. Each character cell value holds the paper colour in bits 0 to 3 and the ink colour in bits 4 to 7.

In multicolour, colours 1 and 2 are held in the colour array with colour 1 in bits 4 to 7 and colour 2 in bits 0 to 3. Colour 0 is held in register 53281 and colour 3 is held in the colour matrix. The colours are displayed by the bit combinations:

Colour 0...0 0  
Colour 1...0 1  
Colour 2...1 0  
Colour 3...1 1

Since two pixels form a dot in multicolour mode, the horizontal resolution is limited to 160 points.

Any given pixel in the bit map is turned on with the following equations: Assuming that the pixel coordinates are X and Y and the bit map starts at the address BASE.

```
ROW = INT(Y/8)
CHAR = INT(X/8)
LINE = Y AND 7
BIT = 7-(X AND 7)
BYTE = BASE + ROW * 320 +
CHAR*8 + LINE
POKE BYTE, PEEK(BYTE) OR
2 ^ BIT
```

Using bit mapped mode from Basic is slow and over complex. Listing 4 gives a machine code package for handling bit mapped mode.

## LISTING 4

```
EE 2000 FORL=OT045: CX=0: FORD=OT
015: READA: CX=CX+A: POKE49152+
L*16+D, A: NEXID
B2 2010 READA: IFA<>CX THEN PRINT"
ERROR IN LINE": 2040+(L*10): S
TOP
OF 2020 NEXID: END
7E 2040 DATA76, 26, 192, 76, 178, 19
2, 76, 16, 193, 76, 178, 194, 32, 25
3, 174, 32, 1964
7A 2050 DATA138, 173, 32, 247, 183,
165, 20, 164, 21, 96, 32, 12, 192, 1
41, 136, 3, 1755
FC 2060 DATA32, 12, 192, 141, 132, 3
, 32, 12, 192, 141, 133, 3, 173, 136
, 3, 240, 1577
37 2070 DATA20, 32, 12, 192, 141, 13
4, 3, 32, 12, 192, 141, 135, 3, 173,
136, 3, 1361
9C 2080 DATA240, 3, 76, 187, 194, 17
3, 132, 3, 10, 10, 10, 10, 13, 133, 3
, 141, 1338
```

```
EO 2090 DATA137, 3, 32, 133, 192, 32
, 218, 192, 173, 2, 221, 9, 3, 141, 2
, 221, 1711
CE 2100 DATA173, 0, 221, 41, 252, 9,
0, 141, 0, 221, 173, 17, 208, 9, 32,
141, 1638
3E 2110 DATA17, 208, 169, 121, 141,
24, 208, 173, 136, 3, 240, 8, 173, 2
2, 208, 9, 1860
6B 2120 DATA16, 141, 22, 208, 96, 16
9, 220, 133, 169, 169, 0, 133, 168,
32, 0, 193, 1869
DO 2130 DATA173, 137, 3, 162, 8, 160
, 127, 145, 168, 136, 16, 251, 72, 2
4, 165, 168, 1915
E1 2140 DATA105, 128, 133, 168, 169
, 0, 101, 169, 133, 169, 104, 202, 2
08, 231, 32, 8, 2060
C1 2150 DATA193, 96, 173, 2, 221, 9,
3, 141, 2, 221, 173, 0, 221, 41, 252
, 9, 1757
29 2160 DATA3, 141, 0, 221, 173, 17,
208, 41, 223, 141, 17, 208, 169, 21
, 141, 24, 1748
3F 2170 DATA208, 173, 22, 208, 41, 2
39, 141, 22, 208, 96, 169, 224, 133
, 169, 169, 0, 2222
70 2180 DATA133, 168, 169, 0, 162, 6
4, 160, 127, 145, 168, 136, 16, 251
, 72, 24, 165, 1960
D2 2190 DATA168, 105, 128, 133, 168
, 169, 0, 101, 169, 133, 169, 104, 2
02, 208, 231, 96, 2284
5A 2200 DATA120, 165, 1, 41, 249, 13
3, 1, 96, 165, 1, 9, 6, 133, 1, 88, 96
, 1305
A3 2210 DATA32, 12, 192, 141, 138, 3
, 140, 139, 3, 32, 12, 192, 141, 147
, 3, 32, 1359
FD 2220 DATA12, 192, 141, 140, 3, 17
3, 136, 3, 240, 3, 76, 235, 193, 32,
77, 193, 1849
20 2230 DATA32, 133, 194, 32, 0, 193
, 173, 147, 3, 160, 0, 174, 140, 3, 2
40, 6, 1630
55 2240 DATA32, 166, 194, 76, 73, 19
3, 32, 171, 194, 32, 8, 193, 96, 173
, 147, 3, 1783
03 2250 DATA74, 74, 74, 141, 141, 3,
173, 139, 3, 74, 173, 138, 3, 106, 7
4, 74, 1464
8B 2260 DATA141, 142, 3, 173, 147, 3
, 41, 7, 141, 145, 3, 173, 141, 3, 14
1, 143, 1547
01 2270 DATA3, 169, 0, 141, 144, 3, 1
62, 6, 32, 220, 193, 202, 208, 250,
173, 144, 2050
27 2280 DATA3, 133, 171, 173, 143, 3
, 133, 170, 32, 220, 193, 32, 220, 1
93, 24, 173, 2016
2B 2290 DATA143, 3, 101, 170, 133, 1
70, 173, 144, 3, 101, 171, 133, 171
, 169, 0, 141, 1926
82 2300 DATA144, 3, 173, 142, 3, 141
, 143, 3, 32, 220, 193, 32, 220, 193
, 32, 220, 1894
90 2310 DATA193, 24, 173, 143, 3, 10
1, 170, 133, 170, 173, 144, 3, 101,
171, 133, 171, 2006
4B 2320 DATA24, 173, 145, 3, 101, 17
0, 133, 170, 169, 0, 101, 171, 133,
171, 24, 169, 1857
E9 2330 DATA0, 101, 170, 133, 170, 1
69, 224, 101, 171, 133, 171, 96, 16
9, 0, 14, 144, 1966
BC 2340 DATA3, 14, 143, 3, 109, 144,
3, 141, 144, 3, 96, 14, 138, 3, 46, 1
```

```
39, 1143
B5 2350 DATA3, 173, 138, 3, 141, 148
, 3, 173, 139, 3, 141, 149, 3, 173, 1
47, 3, 1540
54 2360 DATA141, 150, 3, 173, 140, 3
, 240, 19, 201, 1, 240, 26, 201, 2, 2
40, 33, 1813
72 2370 DATA169, 1, 32, 60, 194, 169
, 1, 32, 60, 194, 96, 169, 0, 32, 60,
194, 1463
FE 2380 DATA169, 0, 32, 60, 194, 96,
169, 0, 32, 60, 194, 169, 1, 32, 60,
194, 1462
A0 2390 DATA96, 169, 1, 32, 60, 194,
169, 0, 32, 60, 194, 96, 141, 151, 3
, 173, 1571
C6 2400 DATA148, 3, 141, 138, 3, 173
, 149, 3, 141, 139, 3, 173, 150, 3, 1
41, 147, 1655
F0 2410 DATA3, 32, 77, 193, 32, 133,
194, 32, 0, 193, 173, 147, 3, 160, 0
, 174, 1546
95 2420 DATA151, 3, 240, 6, 32, 166,
194, 76, 109, 194, 32, 171, 194, 32
, 0, 193, 1793
B9 2430 DATA24, 173, 148, 3, 105, 1,
141, 148, 3, 173, 149, 3, 105, 0, 14
1, 149, 1466
B6 2440 DATA3, 32, 8, 193, 96, 173, 1
38, 3, 41, 7, 141, 146, 3, 56, 169, 7
, 1216
BC 2450 DATA237, 146, 3, 141, 146, 3
, 24, 169, 1, 174, 146, 3, 240, 4, 10
, 202, 1649
OE 2460 DATA208, 252, 141, 147, 3, 9
6, 17, 170, 145, 170, 96, 73, 255, 4
9, 170, 145, 2137
D6 2470 DATA170, 96, 32, 12, 192, 14
1, 136, 3, 76, 88, 192, 173, 132, 3,
141, 33, 1620
F1 2480 DATA208, 173, 135, 3, 141, 1
34, 2, 169, 147, 32, 210, 255, 173,
133, 3, 10, 1928
10 2490 DATA10, 10, 10, 13, 134, 3, 7
6, 79, 192, 191, 0, 191, 0, 191, 0, 1
91, 1291
```

This code starts at 49152 and has three routines.

### 1. Activate bit map

This clears the bit map, set up the colours and turns on bit map mode. It has two forms:

**High resolution mode:** SYS 49152,0,C1,C2.

C1 = paper colour, C2 = ink colour.

**Multicolour mode:** SYS 49152,1,C0,C1,C2,C3.

C0 = paper colour

### 2. Return to text mode.

This returns you to the normal text screen at its normal position, SYS 49155.

### 3. Draw point

This draws the points at X,Y with the specified pen: SYS 49158,X,Y,PEN where:

PEN = 0 draws the point in paper



colour, i.e. it erases the point.

PEN = 1 draws the point in ink 1.

PEN = 2 draws the point in ink 2.

PEN = 3 draws the point in ink 3.

In high resolution mode, X must be in the range 0 to 319. In multicolour mode, X must be in the range 0 to 159. In either mode, Y must be in the range 0 to 199.

In order to keep the routine as short as possible, I have omitted any range checking of the co-ordinates. If you use values outside the allowed range, a crash may occur.

## 4. Turn on bit map.

Without clearing it: SYS 49161, MODE. MODE = 0, high resolution. MODE = 1, multicolour.

So that you don't lose any memory for Basic, the bit map is placed behind the Kernal ROM and interface chip.

Listing 5 is a simple demonstration.

### LISTING 5

```
A0 10 SA=12*4096
B4 20 SYS SA,1,15,11,12,0
B2 30 X1=5:Y1=5:X2=30:Y2=30:PA=
1:GOSUB1000
B9 40 X1=10:Y1=10:X2=20:Y2=20:P
A=2:GOSUB1000
13 50 X1=27:Y1=1:X2=35:Y2=50:PA
=3:GOSUB1000
C6 60 FOR X = 0 TO 50
24 70 SYS SA+6,X+0,X,RND(1)*4:N
EXT X
FA 80 O=0+1:IF O<10 THEN G0
DA 90 END
30 1000 FOR Y=Y1 TO Y2
79 1010 FOR X=X1 TO X2
40 1020 SYS SA+6,X,Y,PA
45 1030 NEXT X,Y
9A 1040 RETURN
```

## Sprites

Sprites are probably the thing which makes games writer's lives simplest. To those of you who don't know, a spritate is a moveable block of 504 pixels arranged in a block of 21 rows of 24. The design is stored in a similar way to characters in that each row can be represented by 3 bytes with the whole design occupying 63 bytes. These designs are stored as a sequence of blocks in any given bank. The address of any given sprite block is given by:

ADDRESS = (BANK\*16384) + (BLOCK NO\*64)

## Specifying a sprite design

The next step is to tell the VIC which pattern block is to be used. A maximum of eight sprites are possible and each has a pointer. These pointers are located above the screen memory and can be found by:

POINTER ADDRESS = SCREEN ADDRESS + 1014 + SPRITE NO

where SPRITE NO is from 0 to 7

A power up, the screen sits at 1024 so the pointer for sprite 3 is at 1024+1014+3 or 2043. To tell the VIC which pattern to use, you simply POKE the block number into the pointer, eg to set sprite 1 to pattern 43:

POKE 2041, 43

## Turning on a Sprite

Whether or not a sprite is visible is determined by VIC register 53269. Each bit of this register controls a sprite. To activate sprite SP use:

POKE 53269, PEEK(53269) OR (2 ^ SP)

To turn off sprite SN use:

POKE 53269, PEEK(53269) AND 255-2 ^ SP

## Expanded Sprites

Sprites can be expanded in both directions to give four possible sizes. These are controlled by two registers. To expand sprite SP in the X direction use:

POKE 53277, PEEK(53277) OR (2 ^ SP)

To reduce it again use:

POKE 53277, PEEK(53277) AND (255-2 ^ SP)

To expand sprite SP in the Y direction use:

POKE 53271, PEEK(53271) OR (2 ^ SP)

To reduce it again use:

POKE 53271, PEEK(53271) AND (255-2 ^ SP)

## Colours

Each sprite has a colour register. This is given by:

REGISTER = 53287 + SPRITE NO

This is used to specify the colour of high resolution sprites

In multicolour sprites the colours are selected by the usual bit pairs, see Table 3.

The eight bits in register 53276 control the colour mode.

To set a sprite SP to multicolour mode use:

POKE 53276, PEEK(53276) OR (2 ^ SP)

To set sprite SP to high resolution mode use:

POKE 53276, PEEK(53276) AND (255-2 ^ SP)

## Positioning a Sprite

The positioning of any given sprite on the screen is defined by its X,Y co-ordinates. The X co-ordinate can range from 0 to 512 and Y co-ordinate from 0 to 256. Each sprite has a dedicated pair of registers. The first holds part of the X position and the other holds the Y co-ordinates. They can be found from:

X Register = 53248 + SN\*2

and the Y register is found from:

Y Register = 53249 + SN\*2

The X position is defined in two parts:

Most significant byte (msb) = INT(XPOS/256)

Least significant byte (lsb) = XPOS-msb\*256

Register 53264 holds the msb details, one bit per sprite

So to position a sprite you use:

POKE XREG,LSB  
POKE YREG,Y

If msb=1 then POKE 53264, PEEK(53264) OR 2 ^ SP.

If msb=0 then POKE 53264, PEEK(53264) AND (255-2 ^ SP).

Table 4.

BIT PAIR	COLOUR SOURCE
0 0	Screen colour
0-1	Register 53285
1 0	Colour register
1 1	Register 53286

Selecting Colour Mode

### Priorities

Each sprite has a priority which decides whether it appears in front of or behind the characters on the screen. Register 53275 decides this, one bit per sprite

To put sprite SP behind the characters use:

POKE 53275, PEEK(53275) OR (2 ^ SP)

To put sprite SP in front of the characters use:

POKE 53275, PEEK(53275) AND (255-2 ^ SP)

That's quite a mouthful and hardly conducive to simple programming. Listing 6 gives the ubiquitous machine code package.

This code has four routines.

### Setup Sprite

SYS 49408,SP,TYPE,COLOUR,XEXP,PRIORITY,(COLOUR1, COLOUR2)

where: SP sprite number (0 to 7).  
TYPE=0=High resolution, 1=Multi-colour.

COLOUR - High resolution colour.  
XEXP - 1= X direction, 0=don't expand X direction.

YEXP - 1=expand Y direction, 0=don't expand Y direction.

PRIORITY=1=behind background, 0=in front of background.

COLOUR1 - Multicolour 1, only needed if TYPE =1.

COLOUR2 - Multicolour 2, only

needed if TYPE=1.

### Switch on

SYS 49411,SP,FLAG where:

FLAG=1 - turn on sprite SP.

FLAG=0 - turn off sprite SP.

### Sprite position

SYS 49414,SP,X,Y where:

SP - sprite number.

X - X position.

Y - Y position.

### Pattern

SYS 49417,SP,DESIGN, BLOCK

The routine is quite smart in that it sorts out which bank you are using and where the sprite pointers are. I therefore recommend that you use the configuration used earlier (screen at 50176 and characters at 51200. This allows you a block of 128 sprites from design block 128 to 255.

### In Summary

In all, this has been a hefty slab of information and I must apologise for not giving more detail. If you want to really get into graphics you must invest in the Programmers Reference Guide or something similar. Having said that, I believe that the routines I've given will be useful tools.

62	2000 FORL=OTD30: CX=0: FORD=OT	F4	2130 DATA172,133,3,185,50,19	60	2240 DATA27,208,96,32,159,19
	D15: READA: CX=CX+A: POKE49408+		3,13,28,208,141,28,208,96,17		4,169,0,141,142,3,32,95,194,
	L*16+D,A: NEXTD		2,133,3,1766		172,142,1806
82	2010 READA: IFA<>CX THEN PRINT"	68	2140 DATA185,58,193,45,28,20	6C	2250 DATA3,145,253,200,192,6
	ERROR IN LINE";2040+(L*10):S		8,141,28,208,96,32,134,194,1		4,240,6,140,142,3,76,75,194,
	TOP		65,20,41,1776		96,169,1998
OF	2020 NEXTL: END	08	2150 DATA1,240,15,172,133,3,	15	2260 DATA255,141,14,212,141,
F1	2040 DATA76,194,194,76,12,19		185,50,193,13,29,208,141,29,		15,212,169,128,141,18,212,16
	3,76,66,193,76,241,193,32,14		208,76,1696		9,128,141,24,2120
	4,194,32,1992	57	2160 DATA206,193,172,133,3,1	AG	2270 DATA12,173,27,212,96,3
C3	2050 DATA134,194,165,20,41,1		85,58,193,45,29,208,141,29,2		2,134,194,165,20,141,37,208,
	,240,13,172,133,3,185,50,193		08,32,134,1969		32,134,194,2011
	,13,21,1578	BB	2170 DATA194,165,20,41,1,240	2D	2280 DATA165,20,141,38,208,9
FO	2060 DATA208,141,21,208,96,1		,13,172,133,3,185,50,193,13,		6,32,253,174,32,138,173,32,2
	72,133,3,185,58,193,45,21,20		23,208,1654		47,183,96,2028
	8,141,21,1854	1F	2180 DATA141,23,208,96,172,1	9B	2290 DATA32,134,194,165,20,2
AD	2070 DATA208,96,1,2,4,8,16,3		33,3,185,58,193,45,23,208,14		01,8,144,2,169,7,141,133,3,9
	2,64,128,254,253,251,247,239		1,23,208,1860		6,32,1481
	,223,2026	CA	2190 DATA96,32,144,194,32,13	BD	2300 DATA134,194,165,20,133,
C3	2080 DATA191,127,32,144,194,		4,194,165,20,133,251,173,136		253,169,0,133,254,162,6,6,25
	32,134,194,165,20,164,21,141		,2,24,105,1835		3,38,254,2174
	,134,3,140,1836	D3	2200 DATA4,133,254,169,0,133	5D	2310 DATA202,208,249,173,0,2
3E	2090 DATA135,3,32,134,194,16		,253,56,165,253,233,8,133,25		21,41,3,168,185,28,194,24,10
	5,20,141,136,3,173,133,3,24,		3,165,254,2466		1,254,133,2184
	10,168,1474	F1	2210 DATA233,0,133,254,172,1	09	2320 DATA254,96,32,144,194,3
B3	2100 DATA173,134,3,153,0,208		33,3,165,251,145,253,96,192,		2,134,194,165,20,141,141,3,3
	,173,136,3,200,153,0,208,173		128,64,0,2222		2,140,193,1915
	,135,3,1855	85	2220 DATA32,134,194,165,20,4	7F	2330 DATA32,134,194,165,20,1
D1	2110 DATA240,13,172,133,3,18		1,1,240,13,172,133,3,185,50,		72,133,3,153,39,208,32,170,1
	5,50,193,13,16,208,141,16,20		193,13,1589		93,32,32,1712
	8,96,172,1859	54	2230 DATA27,208,141,27,208,9	52	2340 DATA194,173,141,3,240,3
3B	2120 DATA133,3,185,58,193,45		6,172,133,3,185,58,193,45,27		,32,117,194,96,133,3,96,32,2
					28,255,1940



# Swapper 64

*Use part of your C64's memory as a RAM disk for storing Basic programs.*

**S**wapper 64 is a utility which allows a Basic program resident in memory to be stored in RAM for recall later, rather like a RAM-disk. Commands are available for storing and recalling a program. Swapping the program in Basic memory for that in storage, and storing part of a program (linenumber to linenumber).

The program can be used either in direct mode or in program mode, the latter being especially useful for utilising two programs held concurrently in memory – one in basic memory and one in storage. These can be rapidly interchanged when desired via the swap command.

## Using the program

Upon initialising the program, the top of Basic memory is lowered to prevent a program overwriting the storage area. This leaves the user with 19K of Basic RAM which should be enough for most programs.

Swapper 64 has five commands which are activated by the syntax:

SYS 49152,command number  
(1-5),condition 1, condition 2

The commands are as follows:

### 1 – Initialise

#### SYS 49152,1

This lowers the top of Basic memory to \$5400 to prevent overwriting the storage area. This should always be the first command executed when Swapper 64 is to be used.

### 2 – Store

#### SYS 49152,2

This command stores the resident Basic program into memory. this can be recalled using command three or swapped with another program using command four.

### 3 – Recall

SYS 49152,3 – recall program  
SYS 49152,3,1 – recall program and auto-run

This function recalls a program in storage, overwriting any program currently in basic memory. If a one is added to the SYS statement, the recalled program will auto-RUN. Make sure that there is a program in storage before you call this command.

Once recalled the storage area is NOT cleared. This means that a program can be recalled more than once if accidently NEWed.

### 4 – Swap

SYS 49152,4 – swap Basic program with stored program  
SYS 49152,4,1 swap Basic program with stored program and auto-run.

Using this command swaps the program in basic memory with a program stored in RAM. As in command three, if a one is added to the SYS instruction, the recalled program will auto-run. Again, ensure that there is a program in stored memory before calling this instruction.

### 5 – Store part of a program

SYS 49152, beginning line, end line.

This stores only the part of the program specified by the beginning line and end line numbers stated in the SYS statement. This commands main use is for when two programs are to be set up in memory to utilise the swap command. The programmer can skip the initial lines of the first program stored ( these are the lines that LOAD in the next part) so that they are not re-run every time this program is re-called from memory.

For example the first program LOAded would have initial lines something like those in Figure 1. The lines have the following function:

By Chris Horton

10 If swapper has been LOADED then skip lines 20 and 30

20 Set A to show that SWAPPER will have been LOADED

30 LOAD SWAPPER into memory

40 Initialise memory

50 SAVE the part of PROG1 that is wanted

60 LOAD next program

10 IF A=1 THEN 40

20 A=1

30 LOAD "SWAPPER64",1 (or ',8' for disk),1

40 SYS 49152,1

50 SYS 49152,5,70,end line of program

60 LOAD "PROG2",1 (or ',8' for disk)

70 REM \*START OF PROGRAM 1\*

Figure 1

PROGRAM: SWAPPER LDR

```

19 10 BL=50:LN=70:SA=49152
F9 20 FOR L=0 TO BL:CX=0:FOR D=
    0 TO 15
9F 30 READ A:IF A>255THENPRINT"
    NUMBER TO LARGE":LN+(L*10):S
    TOP
28 40 CX=CX+A:POKE SA+L*16+D,A:
    NEXT D
A9 50 READ A:IF A<CX THENPRINT
    "ERROR IN LINE":LN+(L*10):ST
    OP
4C 60 NEXT L:END
F1 70 DATA 32,253,174,32,138,17
    3,32,247,183,165,21,208,44,1
    65,20,240,2127
64 80 DATA 40,201,6,176,36,201,
    1,240,23,201,2,240,13,201,3,
    240,1824
64 90 DATA 12,201,4,240,14,201,
    5,240,13,0,76,62,192,76,87,1
    92,1615
C4 100 DATA 76,71,193,76,80,193
    ,76,228,193,162,11,76,55,164
    ,32,118,1804
C0 110 DATA 192,160,0,177,251,1
    45,253,76,145,192,162,0,181,
    43,157,59,2193
A4 120 DATA 195,232,224,14,208,
    246,96,32,20,193,32,105,192,
    32,118,192,2131
A6 130 DATA 160,0,177,253,145,2
    51,76,180,192,162,0,189,59,1
    95,149,43,2231
52 140 DATA 232,224,14,208,246,
    96,169,0,133,251,169,8,133,2
    52,165,45,2345
A1 150 DATA 141,39,195,165,46,1
    41,40,195,169,0,133,253,169,
    84,133,254,2157
AB 160 DATA 96,230,251,208,2,23
    0,252,230,253,208,2,230,254,
    165,252,205,3068
B3 170 DATA 40,195,240,3,76,67,
    192,165,251,205,39,195,240,3
    ,76,67,2054
5B 180 DATA 192,76,74,192,230,2
    51,208,2,230,252,230,253,208
    ,2,230,254,2884
37 190 DATA 165,252,205,40,195,
    240,3,76,98,192,165,251,205,
    39,195,240,2561
2F 200 DATA 3,76,98,192,165,43,

```

```

133,251,165,44,133,252,32,22
    ,195,160,1964
OB 210 DATA 0,169,0,145,251,76,
    59,193,230,251,208,2,230,252
    ,230,253,2549
BC 220 DATA 208,2,230,254,165,2
    52,201,84,240,3,76,137,193,3
    2,102,193,2372
E1 230 DATA 165,43,133,251,165,
    44,133,252,32,22,195,160,0,1
    69,0,145,1909
22 240 DATA 251,76,59,193,169,4
    4,160,0,209,122,240,6,169,0,
    141,37,1876
FA 250 DATA 195,96,32,253,174,3
    2,138,173,32,247,183,165,21,
    208,237,165,2351
D1 260 DATA 20,201,1,208,231,16
    9,1,141,37,195,96,173,37,195
    ,208,1,1914
23 270 DATA 96,32,89,166,76,174
    ,167,169,255,133,55,169,83,1
    33,56,96,1949
FA 280 DATA 32,20,193,169,0,133
    ,251,169,8,133,252,169,0,133
    ,253,169,2084
8F 290 DATA 84,133,254,76,126,1
    93,162,0,181,43,141,41,195,1
    89,59,195,2072
B6 300 DATA 149,43,173,41,195,1
    57,59,195,232,224,14,208,235
    ,96,160,0,2181
88 310 DATA 140,35,195,140,34,1
    95,140,36,195,177,251,141,41
    ,195,177,253,2345
B1 320 DATA 145,251,201,0,208,1
    1,173,34,195,208,6,238,35,19
    5,76,164,2140
04 330 DATA 193,140,35,195,173,
    41,195,145,253,173,35,195,20
    1,3,240,21,2238
48 340 DATA 165,252,197,46,208,
    12,165,251,197,45,208,6,76,2
    15,193,76,2312
0D 350 DATA 253,192,76,232,192,
    169,1,141,34,195,238,36,195,
    173,36,195,2358
96 360 DATA 201,2,240,235,76,17
    6,193,238,36,195,173,36,195,
    201,2,240,2439
53 370 DATA 222,76,194,193,32,2
    53,174,32,138,173,32,247,183
    ,165,20,141,2275
31 380 DATA 42,195,165,21,141,4
    3,195,32,253,174,32,138,173,
    32,247,183,2066

```

```

98 390 DATA 165,20,141,44,195,1
    65,21,141,45,195,173,43,195,
    205,45,195,1988
DD 400 DATA 240,25,176,34,165,4
    3,133,251,165,44,133,252,162
    ,0,181,43,2047
E5 410 DATA 157,50,195,232,224,
    5,208,246,76,57,194,173,42,1
    95,205,44,2303
17 420 DATA 195,176,3,76,20,194
    ,76,57,192,173,42,195,141,48
    ,195,173,1956
04 430 DATA 43,195,141,49,195,3
    2,217,194,32,22,195,32,22,19
    5,32,22,1618
1F 440 DATA 195,165,251,141,38,
    195,165,252,141,39,195,173,4
    6,195,133,251,2575
59 450 DATA 173,47,195,133,252,
    173,44,195,141,48,195,173,45
    ,195,141,49,2199
E3 460 DATA 195,32,217,194,173,
    46,195,133,251,173,47,195,13
    3,252,165,251,2652
BD 470 DATA 141,55,195,165,252,
    141,56,195,177,251,141,57,19
    5,169,0,145,2335
9A 480 DATA 251,32,15,195,177,2
    51,141,58,195,169,0,145,251,
    32,15,195,2122
28 490 DATA 173,38,195,133,43,1
    73,39,195,133,44,165,251,133
    ,45,165,252,2177
8A 500 DATA 133,46,32,62,192,16
    2,0,189,50,195,149,43,232,22
    4,5,208,1922
C4 510 DATA 246,173,55,195,133,
    251,173,56,195,133,252,160,0
    ,173,57,195,2447
6E 520 DATA 145,251,200,173,58,
    195,145,251,96,160,0,177,251
    ,141,46,195,2484
74 530 DATA 32,15,195,177,251,2
    40,24,141,47,195,32,15,195,1
    77,251,205,2192
E1 540 DATA 48,195,208,14,32,15
    ,195,177,251,205,49,195,208,
    4,96,76,1968
BE 550 DATA 57,192,173,46,195,1
    33,251,173,47,195,133,252,76
    ,219,194,230,2566
B5 560 DATA 251,208,2,230,252,9
    6,166,251,202,134,251,224,25
    5,208,2,198,2930
56 570 DATA 252,96,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,348

```



# 128 Disk Utility

Create auto boot disks and much more with this powerful utility.

**H**ow many times have you wanted to load a program from the directory and been frustrated to see the SYNTAX ERROR message appear when you press RETURN? The reason for this is that the letters PRG still existed on the command line when you pressed the RETURN key. It is very annoying and is the main reason that I decided to write the C128 Disk Utility.

If you wish to LOAD a program directly from the directory listing you must either remove the three characters (usually PRG) representing the filetype, from the line or you could place a colon before them before pressing return. In effect your command line would look like this.

DLOAD "program name" : PRG

This is a valid command. This program will modify any file you wish, to enable you to simply type DLOAD before the file name in the directory listing. It does this by modifying the disk directory and placing a colon outside the quotes but before the filetype flag so that each directory entry will look like the above.

C128 Disk Utility has many other functions as well but the *put colon after filename* feature is probably the most useful. It is well worth the trouble of obtaining a copy of C128 Disk Utility simply to have it available.

## Other Options

C128 Disk Utility also offers the user the following functions:

RENAME FILES  
INITIALIZE THE DISK DRIVE  
DISPLAY DIRECTORY OF DISK  
MAKE A BOOT DISK  
PRINT DIRECTORY IN DETAIL  
DELETE FILES ON DISK  
1541/1571 FORMAT A DISK  
CLEAN UP A DISK  
LOCK A FILE AGAINST SCRATCH

UNLOCK THE FILE AGAIN  
PERFORM A QUICK SCAN AND DELETE  
RECOVER A SCRATCHED FILE  
QUIT THE PROGRAM  
CHANGE SCREEN COLOURS

As you can see, C128 Disk Utility is a very valuable addition to your program library.

Each function of the program is described below.

## RUNNING THE PROGRAM.

C128 Disk Utility is written completely in Basic V7.0, so it is simply LOADED, SAVEed and RUN as any other Basic program. There will be a short delay while the program initialises variables and strings.

## RENAMING FILES.

This is selected from the main menu as are the rest of the functions available.

To select from the menu, move the pointer at the right of the menu block up and down using the cursor up/down keys, until it points to the function that you want.

After you make your selection press RETURN to indicate to the computer that you want to accept the function the arrow is pointing to.

Selecting RENAME FILES will gain entry to a sub-menu. This menu works in exactly the same way as the main menu. You may select from PLACE COLON AFTER FILENAME, RENAME FILES, or GO TO MAIN MENU. Couldn't be simpler right? Should you select the PLACE COLON AFTER FILENAME option you will be asked if you want to perform the operation on all the files on the disk or by selection from the directory. Select the former and the machine will carry out its operation on all the files providing there is sufficient room after the filename for

the necessary change to the directory.

Any errors will be reported and the program always defaults back to the main menu after an error occurs.

If you elect to choose files to be altered there will be a short delay while the program memorizes the directory and you will then be offered the files on the disk one after another and asked if you wish to alter them. You will be able to select one of three possible replies to the prompt, these are "Y" for yes, "N" for no, or "C" for cancel everything and go back to the menu. This holds true for virtually all modes in C128 Disk Utility.

## INITIALIZE DRIVE

This will clear the error channel on the drive and perform a dclear.

## DISPLAY DIRECTORY

Selecting this mode will print the directory on the screen. You may slow the display down by pressing the COMMODORE key, or pause the display by pressing the NO scroll key. Follow the prompt on the screen to return to the main menu.

## MAKE A BOOT DISK

As the name suggests, this allows you to make a disk which is BOOTable on the C128 computer. A BOOTable disk is one which will load a nominated program, either by pressing the reset button, entering the command BOOT (RETURN), or switching the computer on with the disk in the drive. Your C128 manual will explain this further. To create a BOOT disk with C128 Disk Utility you should follow the procedure below.

Select the disk you wish to BOOT. It may have been used before or not. If not it will have to be formatted first. This is therefore the first question you will be asked, FORMAT DISK? (Y/N). Select

By K.R.Sharkey

the appropriate answer. See the part of this article pertaining to formatting disks for information about this.

You now should have formatted in the drive either with or without any programs on it. *C128 Disk Utility* will now perform a check of track one to ensure that there is no chance of writing over any data on the disk. The program will report if it is safe to proceed, at the same time asking for your assurance that you want to continue. If you reply NO there will be no harm done to the disk and you will be returned to the main menu, however, if you wish to continue you will be asked for the name of the program to be BOOTed. Upon answering this question you are asked if the program is a Basic program if you reply NO then the boot will be a non-relocatable boot, otherwise the boot will be for a Basic program.

The BOOT sector will now be written to the disk and you may SAVE the program you wish to boot, if it is not already there.

When you wish to LOAD and RUN the program simply BOOT the disk (see your Commodore manual for more details) and it will come up RUNNING.

## PRINT DIRECTORY

This function prints out a detailed directory onto the printer or the screen. To nominate the screen, switch the printer off and the program will default to the screen.

Only one question needs to be answered in this mode and that is whether or not to show files in the directory which have been scratched. This can be handy if you are looking for a deleted file on a disk which you may be able to recover using *C128 Disk Utility*.

The printout of the directory will show the following information:

DISK NAME  
DISK ID  
DISK FORMAT (usually 2A)  
FILETYPE OF PROGRAM  
TRACK WHERE IT LIVES  
SECTOR ON THAT TRACK  
FILENAME OF PROGRAM  
NUMBER OF BLOCKS USED  
START LOCATION IN C128  
BLOCKS FREE ON DISK

## DELETE FILES

This mode works exactly the same as **RENAME** except it deletes the file rather than renaming it. You are still given the option of which programs to scratch from the disk. If you make a mistake don't panic *C128 Disk Utility* can recover the program, so read on.

## FORMAT A DISK

Any disk to be used on a Commodore C128 must first be formatted. Most of you will have learned this already, but if not please refer to the manual which came with your computer for details.

This section of the program allows you to format a disk without the need to know the necessary syntax. All you have to know is whether you want the disk in 1571 double sided format or 1541 single sided format, and what name you would like to give the disk.

*C128 Disk Utility* takes care of things from now on. If you wish you can even forget about the disk name and ID as the program will give the disk one for you. Any errors occurring during the preparation of the new disk will be reported, and eventually you will be passed back to the main menu.

If you elect not to give a name and ID for the disk and the disk has been used before, the program will do a fast directory clear out on the disk, which only takes a few seconds. Please note that you cannot recover any programs from the disk with *C128 Disk Utility* once a format has been carried out, so be careful.

## COLLECT

Collect is the same as the **COLLECT** command in Basic V7.0. Disks in use should be **COLLECT**ed periodically to ensure that they have a valid BAM (Block Allocation Map) and that they are using up disk space efficiently. A **COLLECT** should always be done after you recover any scratched file using this program.

## LOCK/UNLOCK A FILE

It is not commonly known that it is

possible to put a security lock flag on a file to prevent that file from inadvertently being scratched. This function does just that. When you select this part of the program, you will be passed to the **FILETYPE EDITOR** part of *C128 Disk Utility*. Simply follow the prompts on the screen and you won't go wrong.

If you elect to place a lock on a program, it will appear in the directory as a 'L' symbol beside the filetype.

By selecting unlock (simply press U when you are offered the program you wish to unlock) you can remove the lock.

## QUICK DELETE/RECOVER FILES

Once a program or file has been scratched from the disk it is still possible to recover it providing the area on the disk which the program occupies has not been over written by a subsequent write to the disk. If it is at all possible to recover the file this feature of *C128 Disk Utility* will do it. Select R when the program you wish to recover appears on the screen.

You will now be asked which type of file to recover, and your options are:

- 0=DELETED (not normally used)
- 1=SEQUENTIAL (usually text)
- 2=PROGRAM (normally this one)
- 3=USER (special user design)
- 4=RELATIVE (usually a database)

Select the number corresponding to the filetype you require and the program will be recovered as that type of file. It is always wise to carry out a **COLLECT** of the disk after this is done.

Quick Delete is a much faster way to delete a selected file than the previous Delete function. This is because it doesn't have to read the whole directory before it commences, it also does not clean up the BAM after it is finished.

Should you press the return key at the prompt in the Filetype Editor then no change will occur on the disk and you will be offered the next file in the directory.

## QUIT

Quit does just that. It restores the default Commodore colours and returns control to BASIC.



## CHANGE COLOURS

This has been added to the program just in case you don't like my colour selection. Enter a number between one and sixteen and you will be asked if it is correct. Answer YES or NO and press RETURN. If you selected yes the colour will be transferred to the screen and you will then be asked to select a new border and then a new character colour.

If you answer NO when asked if your selection is correct you will be asked again to enter a new colour.

I hope you find the *C128 Disk Utility* useful in maintaining your disk files.

PROGRAM: 128 DISK UTIL

```

1 REM *****
2 REM *                                     DI
SK UTILITY PROGRAM
3 REM *-----
4 REM *
BY
5 REM *
K. R. SHARKEY
6 REM *-----
7 REM * TO USE SIMPLY RUN THI
S PROGRAM - IT WILL GIVE THE OPT
ION OF
8 REM * EITHER CONVERTING ALL
FILES ON DISK OR SELECTING CERT
AIN FILES
9 REM * THE FILES WILL THEN BE
RENAMED TO ALLOW DLOAD TO BE US
ED IN
10 REM * FRONT OF THEIR NAME O
N THE DIRECTORY LISTING WITHOUT
THE
11 REM * ANNOYANCE OF HAVING I
O INSERT A COLON AFTER THE FILE
NAME OR
12 REM * CLEARING OUT THE REST
OF THE LINE ON THE SCREEN.
13 REM *
14 REM * A BOOT DISK CAN BE CRE
ATED BY SIMPLY SELECTING 'MAKE B
OOT DISK'
15 REM * ON THE MENU AND THEN A

```

ANSWERING THE PROMPTED QUESTIONS.

```

16 REM *
17 REM * PROGRAMS CAN BE RENAME
D OR DELETED ON THE DISK IN THE
DRIVE BY
18 REM * SELECTING THE CORRECT
COMMAND FROM THE MENU & ANSWERIN
G THE PROMPTS.
19 REM *
20 REM * LOCK/UNLOCK WILL ALLOW
YOU TO LOCK A FILE SO THAT IT C
ANNOT BE
21 REM * SCRATCHED OR UNLOCK IT
AGAIN AT WILL. RESTORE RECOVERS
SCRATCHED
22 REM * FILES. NOTE: IT IS WIS
E TO VALIDATE THE BAM AFTER UNSC
RATCHING.
23 REM *

```

```

24 REM * SELECT 'PRINT DIRECTOR
Y' AND A DETAILED DIRECTORY WILL
BE LISTED ON
25 REM * THE PRINTER. 'COLLECT'
WILL CLEAN UP THE BAM ON THE DI
SK.
26 REM *****
27 COLOR RGR(0),1:COLOR4,1:COLOR
5,15:IFRGR(0)=5THENPRINTCHR$(27)
;"R"
28 CLR:DIMS$(15),NAS$(120),NES(12
0):BES=CHR$(7)+CHR$(15)
29 TYS(0)=-"DELETED":TYS(1)=-"S
EQUENTIAL":TYS(2)=-"PROGRAM":T
YS(3)=-"USER":TYS(4)=-"RELAT
IVE"
30 TRAP292:REM ROUTINE TO HANDLE
ERRORS
31:
32 REM ***** DISPLAY MENU *****
33:
34 SCNCLR:FAST:GS-1:YU-15:RESTOR
E366:IF MID$(S$(4),10,1)=-"R"THE
N35:ELSEGOSUB298
35 IFRGR(0)=5THENWINDOW0,0,79,24
:ELSEWINDOW0,0,39,24
36 SLOW:SCNCLR:CHAR1,5,0," M
AIN MENU":1:GOSUB37:GOTO41
37 FORXT=4TOYU:CHAR1,0,XT,S$(XT)
,1:NEXT
38 CHAR1,1,17,"USE CRSR U/D TO S
ELECT"
39 CHAR1,1,19,"PRESS RETURN TO A
CCEPT SELECTION"
40 X=30:Y=6:CHAR1,X,Y,"[RVSON]_C
RVSOFF":RETURN
41 GOSUB54
42:
43 IF GSTHEN ON Y GOSUB,,48,80,
85,91,132,137,142,171,309,309,17
9,372
44 IFGS=0THEN ON Y GOSUB,,61,75
,78:RETURN
45 RETURN
46:
47 REM ***** RENAME FILES *****
48:
49 GOSUB 187:SCNCLR
50 FAST:GS=0:YU=6:RESTORE 368:GO

```

```

SUB298:SLOW
51 CHAR1,5,0," RENAME MENU
",1
52 GOTO36
53 REM ***** MAIN LOOP *****
54 DO:GETAS:LOOP UNTIL AS="[DOWN
]"OR AS="[CUP]"ORAS=CHR$(13)
55 IFAS=CHR$(13)THENGOSUB43:GOTO
29
56 IFAS="[DOWN]"THENPS="-":GOSUB
59:Y=Y+1:PS="[RVSON]_RVSOFF":G
OSUB59
57 IFAS="[CUP]"THENPS="-":GOSUB59
:Y=Y-1:PS="[RVSON]_RVSOFF":GOS
UB59
58 GOTO54: STOP
59 IFY>YUTHE N Y=YU
60 CHAR 1,X,Y,PS:CHAR0,0,Y,S$(Y
):CHAR1,0,Y,S$(Y),1: RETURN
61 PRINT"[CLEAR]RENAME USING COL
ON":PRINT"[DOWN][DOWN]APPLY COLO
N TO ALL NAMES [RVSON](Y/N/C)[RV
SOFF]"
62 DO:GETAS:LOOP UNTIL AS="N" OR
AS="Y" ORAS="C"
63 IF AS="Y"THEN 70
64 IF AS="C"THENPRINT"OK:-CANCEL
LED":SLEEP3:GOTO46
65:
66 REM ***** SELECTED NAMES *****
67:
68 GOSUB 183
69 GOSUB219:SC=0:SC$="WRITE COLO
N AFTER":GOSUB 246:CLOSE1:RETURN
70:
71 REM ***** RENAME ALL *****
72:
73 GOSUB219:GOSUB 258:CLOSE1:RET
URN
74:
75 REM ***** RENAME FILES *****
76:
77 GOSUB219:SC=0:SC$="CHANGE":GO
SUB271
78 CLOSE1:RETURN
79:
80 REM ***** INITIALIZE *****
81:
82 UT$="":PRINT"[CLEAR]";S$(5):C
LOSE15:OPEN15,8,15,"10":CLOSE15
:IFDSTHENGOSUB187:SLEEP4:PRINT
83 RETURN
84:
85 REM ***** DISPLAY DIR *****
86:
87 PRINT"[CLEAR]";S$(6):CHAR1,1,
1,"[RVSON]NO SCROLL[RVSOFF] KEY
TO PAUSE[HOME][DOWN][DOWN][DOWN]
"
88 GOSUB183:DIRECTORY:IFDSTHENPR
INTCHR$(7);CHR$(15);DS$:SLEEP4:E
LSEPRINTCHR$(15);CHR$(7);"[DOWN]
PRESS A KEY":GETKEY AS
89 RETURN
90:
91 REM ***** SETUP BOOT *****
92:
93 PRINT"[CLEAR]";S$(7):NS=CHR$(
0):GOSUB183
94 CHAR1,0,3,"FORMAT DISK [RVSON
](Y/N)[RVSOFF]"
95 DO:GETAS:LOOP UNTILAS="Y"ORAS
="N"
96 IFAS="Y"THENFL$="-":GOSUB141:R

```

```

EM GO DO FORMAT
97 OPEN15,8,15:OPEN8,8,8,"#":PRI
NT#15,"B-A:":0;1;0
98 IF DS=0 THEN PRINT:PRINT"BOOT
SECTOR IS FREE":FR=1:GOTO105
99 FR=0:PRINT#15,"U1":8;0;1;0
100 PRINT#15,"B-P":8;0:GET#8,C$,
B$,M$:IFC$="C"ANDB$="B"ANDM$="M"
THENBEGIN:PRINT#15,"B-P":8;7:PR
INT:RUS=""
101 DO:GET#8,A$:RUS=RUS+A$:LOOP
UNTIL A$=""
102 IFRUS>" "THENRUS="BOOT FOR "+
RUS+" EXISTS"
103 PRINTRUS:BEND
104 PRINT"BOOT SECTOR IS IN USE"
105 PRINT:PRINTCHR$(15);CHR$(7);
"CONTINUE OPERATION [RUSON](Y/N)
[RUSOFF]"
106 DO:GETAS:LOOP UNTILAS="Y"ORA
S="N"
107 IFAS="N"ANDFR=1THENPRINT#15,
"B-F:":0;1;0
108 IFAS="N"THENCLOSE8:CLOSE15:R
ETURN
109 DO
110 SCNCLR:CHAR1,0,24,"ENTER QUI
T TO RETURN TO MENU",1
111 INPUT[HOME][DOWN][DOWN][DO
W][DOWN][DOWN]NAME OF BOOT PROGR
AM":NN$
112 IFLEN(NN$)>16THENPRINTERR$(2
3):NN$="":SLEEP3:GOTO115
113 IFNN$=""THENPRINTERR$(8):SLE
EP3
114 IFNN$="QUIT"THENPRINT"ABORTI
NG":SLEEP3:CLOSE8:CLOSE15:RETURN
115 LOOPWHILE LEN(NN$)>16ORNN$=""
116 PRINT"IS "NN$:PRINT"A BASIC
PROGRAM [RUSON](Y/N)[RUSOFF]"
117 DO:GETAS:LOOP UNTILAS="Y"ORA
S="N"
118 IFAS="N"THENBEGIN
119 PRINT"BOOT IS FOR NON RELOCA
TABLE PROGRAM"
120 MOS="BOOT":BEND:ELSEMOS="RU
N"
121 U0=LEN(NN$)+2831:UL=(U0AND25
5):UH=INT(U0/256)
122 PRINT#15,"U1:8 0 1 0"
123 PRINT#15,"B-P 8 0"
124 PRINT#8,"CBM":N$:N$:N$:N$:N$
$;N$:N$:
125 PRINT#8,CHR$(162);CHR$(UL);C
HR$(160);CHR$(UH);CHR$(76);CHR$(
165);CHR$(175);
126 PRINT#8,MOS;CHR$(34);NN$:N$
127 PRINT#15,"U2:8 0 1 0"
128 CLOSE8:CLOSE15
129 PRINT"FINISHED.....":PRINT"
SAVE ":NN$;" TO THIS DISK ":PRIN
T"AND IT WILL AUTOBOOT"
130 SLEEPS:RETURN
131 :
132 REM **** PRINT DIRECTORY **
**
133 :
134 PRINT[CLEAR];S$(8):PRINT"W
ILL I PRINT [RUSON]DELETED[RUSOF
F] FILES?":DO:GETQAS:LOOP UNTIL
QAS="Y"ORQAS="N"
135 GOSUB183:CLOSE2:DV=4:GOSUB18
9:CLOSE1:CLOSE2:RETURN
136 :
137 REM **** DELETE FILES ****
138 :

```

```

139 GOSUB219:SC=1:SCS="DELETE":G
OSUB 271
140 CLOSE1:RETURN
141 :
142 REM **** FORMAT DISK ****
143 :
144 PRINT[CLEAR];S$(10):IDS=HE
XS(INI(RND(0)*65535)):IDS=MIDS(I
DS,2,2):DCLEAR:DIS="":SES="":I
FDS=0THENBEGIN
145 PRINT[HOME][DOWN];CHR$(27)
;"@THIS DISK HAS BEEN FORMATTED
BEFORE"
146 PRINT[HOME][DOWN][DOWN];CH
R$(27)"@CONTINUE [RUSON](Y/N)[RUS
OFF]":IDS=""
147 DO:GETAS:LOOP UNTILAS="Y"ORA
S="N"
148 IFAS="Y"THENSES=" [RUSON]OLD
ID[RUSOFF]":GOTO150
149 PRINTBES;[HOME][DOWN];CHR$(
27);"@FORMAT CANCELLED":SLEEP3:
GOTO29
150 BEND
151 INPUT"FORMAT IN 1541 OR 1571
MODE 1571[LEFT][LEFT][LEFT][LE
FT][LEFT][LEFT]";DM
152 IFDM=1541THENMS="U0>M0"
153 IFDM=1571THENMS="U0>M1"
154 PRINT[UP]CHR$(27)"@:INPUT
"INPUT NEW DISK NAME DISK#[LEFT
][LEFT][LEFT][LEFT][LEFT][LE
FT]";NM$
155 PRINT[HOME][DOWN];CHR$(27)
;"@PRESS RETURN FOR QUICK DIRECT
ORY":PRINT"CLEAROUT OR ENTER NEW
ID FOR"
156 DO:INPUT"FULL FORMAT ON DISK
":DIS
157 LOOP UNTIL LEN(DIS)<=20RASC
(DIS)=0
158 IFDIS=CHR$(13)THEN160
159 IFDIS<>" "THENIDS=DIS:SES="
"
160 PRINT[HOME][DOWN];CHR$(27)
;"@FORMATTING :-"NM$;SES;IDS
161 OPEN15,8,15:PRINT#15,M$:PRIN
T#15,"N0:"+NM$+" "+IDS
162 PRINT#15,"U0>M1":CLOSE15
163 PRINT[HOME][DOWN];CHR$(27)
;"@DISK IS READY FOR USE"
164 IFDS<>0THENBEGIN:PRINT[HOME
][DOWN];CHR$(27)"@DISK ERROR OC
CURRED"
165 PRINTCHR$(15);CHR$(7);DS$
166 PRINT"TRY AGAIN [RUSON](Y/N)
[RUSOFF]"
167 DO:GETAS:LOOP UNTILAS="Y"ORA
S="N":IFAS="N"THENPRINT"ABORTING
OPERATION"
168 IFAS="Y"THENPRINT"OK":BEND:G
OTO160
169 SLEEP3:RETURN
170 :
171 REM **** VALIDATE DISK ****
172 PRINTCHR$(7);:SCNCLR:PRINT
S$(11)
173 PRINTCHR$(15);[DOWN]COLLECT
ING A DISK CAN DESTROY"
174 PRINTCHR$(15);"DIRECT ACCESS
FILES:- ":PRINT"CONTINUE [RUSO
N](Y/N)[RUSOFF]"
175 DO:GET AS:LOOP UNTILAS="Y"OR
AS="N"
176 IF AS="Y"THENPRINT[UP][UP][
UP];CHR$(27)"@OK: VALIDATING D
ISK":COLLECT:FL$="":GOSUB187:SLE

```

```

EP3:DCLEAR:RETURN
177 PRINT[CUP][CUP][CUP][CUP];CHR$
(27);"@OK:- CANCELLED":SLEEP3:RE
TURN
178 :
179 REM **** END PROGRAM ****
180 :
181 SCNCLR:PRINT"END PROGRAM :-
ARE YOU SURE?":DO:GETAS:LOOP UNT
IL AS="Y"ORAS="N"
182 IFAS="Y"THENCOLOR0,12:COLOR
4,14:COLOR5,14:CLR:GRAPHIC RGR(
0),1:GRAPHICCLR:NEW:SYS16384:ELS
E RETURN
183 :
184 REM **** DCLEAR & CHECK ERRO
RS
185 :
186 DCLEAR
187 PES=BES+"DISK ERROR :-"+DS$:
IF DS THENCHAR1,0,24,PES,1:SLEEP
3:PRINT[CLEAR]:GOTO29
188 RETURN
189 :
190 REM **** GET DIRECTORY ****
191 :
192 PR$="":OPEN2,DV:PRINT#2,"DI
SK NAME:- ":IF -128 AND ST TH
EN PES=BES+ERR$(5):CHAR1,0,24,PE
$,1:SLEEP2:SCNCLR:DV=3:CLOSE2:GO
TO192
193 OPEN15,8,15:OPEN5,8,5,"#":IF
DSTHENPRINTCHR$(7);CHR$(15);DS$:
SLEEP4:RETURN
194 PRINT#15,"U1":5;0;18;0:PRINT
#15,"B-P":5;144
195 DO:GET#5,A$:PRINT#2,A$;:LOOP
UNTIL AS=CHR$(160):DO:GET#5,A$:
LOOP UNTILAS<>CHR$(160)
196 PRINT#2," ID:-":A$;:DO:GET#5
,A$:PRINT#2,A$;:LOOP UNTIL AS=CH
R$(160)
197 PRINT#2,"FORMAT:-":DO:GET#5
,A$:PRINT#2,A$;:LOOP UNTIL AS=CH
R$(160):PRINT#2:FORI=1TO80:PRINT
#2,"[s *]";:NEXT
198 PRINT#2,"FILETYPE TRACK SECT
OR"," NAME BLOCKS
LOCATION":FORI=1TO80:PRINT#2,
"[s *]";:NEXT
199 TR=18:SE=1:PT=TR:PS=SE
200 PRINT#15,"U1":5;0;TR;SE
201 PRINT#15,"B-P":5,0:GET#5,TR$
,SE$:TR=ASC(TR$):SE=ASC(SE$)
202 B=2:FORN=1TO8:PRINT#15,"B-P"
;5;B
203 GET#5,A$:Z=ASC(A$):IFZ<>0AND
Z-192>0THENZ=Z-192:ELSEIFZ<>0THE
N2=Z-128
204 IFQAS="N"ANDZ=0THENGOTO212:E
LSE PR$=PR$+TY$(Z)+" "
205 GET#5,A$:TK=ASC(A$):IFTK=0TH
EN214:ELSEIFTK>9THENPR$=PR$+STR$
(TK):ELSEPR$=PR$+" "+STR$(TK)
206 GET#5,A$:SK=ASC(A$):IFSK>9TH
ENPR$=PR$+STR$(SK):ELSEPR$=PR$+"
"+STR$(SK)
207 PR$=PR$+" ":FORI=
3TO18:GET#5,A$:PR$=PR$+A$:NEXTI:
208 FORI=19TO28:GET#5,A$:NEXT:LO
=ASC(A$):GET#5,A$:HI=ASC(A$):I=L
O+256*HI
209 IFI<=9THENPR$=PR$+" "+STR$(
I):ELSEIFI<=99THENPR$=PR$+" "+ST
R$(I):ELSEPR$=PR$+STR$(I)
210 IFZ<>0THENPRINT#15,"U1":5;0;
TK;SK:PRINT#15,"B-P":5;2:GET#5,A

```



```

$ : LO=ASC(A$):GET#5,A$:HI=ASC(A$)
:Q=LO+256*HI:PR$=PR$+"
"+STR$(Q)
211 PRINT#2,PR$
212 PRINT#15,"U1";5;0;PT;PS:B=B+
32:PR$="":NEXIN
213 IFASC(TR$)<>0THENPT=TR:PS=SE
:GOTO200
214 FORJ=1TO80:PRINT#2,"[s*]";:
NEXIJ:PRINT#2:CLOSE5
215 NU$=CHR$(0):OPEN1,8,0,"$0:$%
&'":GET#1,A$,A$,A$,A$,A$,A$
216 GET#1,A$:IFAS<>"":THEN216
217 GET#1,A$,A$,A$,B$:BC=ASC(A$+
NU$)+ASC(B$+NU$)*256
218 PRINT#2,CHR$(27)"-CHR$(1)"*
***;BC: "**** BLOCKS AVAILABLE "C
HR$(27)"-CHR$(0):CLOSE1:CLOSE15
:SLOW:IFDV=3THENPRINT"PRESS A KE
Y":DO:GETAS:LOOPWHILEAS="":REUR
N
219 :
220 REM **** GET NAMES ****
221 :
222 OPEN15,8,15,"I0":PRINT#15,"M
-R"CHR$(18)CHR$(0)CHR$(2):UT$=""
223 FORI=1TO2:GET#15,A$:UT$=UT$+
A$:NEXT
224 IFUT$=FL$THENCLOSE15:RETURN
225 FL$=UT$:CLOSE15
226 FAST:FORXT=0TO120:NAS(XT)=""
:NEXT
227 OPEN1,8,0,"$0":IFDSTHENSLOW:
GOSUB187:SLEEP4:RETURN
228 CO=0
229 SCNCLR:GET#1,A$,A$
230 GET#1,A$,A$:IFAS="":THEN SLOW
:RETURN
231 GET#1,A$,B$
232 GET#1,A$:IFAS="":THENGOTO230
233 IFAS=CHR$(34)THENBEGIN:CO=CO
+1
234 AS="":DO UNTIL AS=CHR$(34):G
ET#1,A$:IFAS<>CHR$(34)THEN NAS(C
O)=NAS(CO)+A$
235 LOOP:BEND
236 GOTO232
237 SLOW:RETURN
238 :
239 REM **** GET BYTES FROM DISK
****
240 :
241 SCNCLR:GET#1,A$,A$
242 GET#1,A$,A$:IFAS="":THENGOTO3
4
243 GET#1,A$,B$
244 GET#1,A$:IFAS="":THENGOTO242
245 RETURN
246 :
247 REM **** SELECT NAMES ****
248 :
249 GOSUB 302
250 FORXT=2TOCO:PRINT"[HOME][DOW
N][DOWN][DOWN]"
251 PRINT"[HOME][DOWN][DOWN][DOW
N][DOWN]";CHR$(27);"@RENAME :-";
NAS(XT);"[RVSON](Y/N/C)[RVSOFF]"
252 DO:GET AS:LOOP UNTILAS="Y"OR
AS="N"ORAS="C":IFAS="C"THENPRINT
"OK:- CANCELLED":SLEEP3:GOTO257
253 IFAS="Y"THENBEGIN:NE$(XT)=LE
FT$(NAS(XT),14)+CHR$(160)+":":FL
$=""
254 RENAME D0,(NAS(XT)) TO (NE$(
XT)),UB :BEND

```

```

255 IFDSTHENPRINTCHR$(7);CHR$(15
):CHAR1,0,24,DS$,1:SLEEP2:GOSUB1
83:PRINT"[UP]"CHR$(27)"@"
256 NEXT
257 RETURN
258 :
259 REM **** CHANGE ALL NAMES **
**
260 :
261 SCNCLR:IF CO=1THEN:PRINTCHR$(
7);CHR$(15)"NO FILES ON THIS DI
SK":SLEEP3:RETURN
262 PRINT"CURRENT DISK :-[RVSON]
";NAS(1);"[RVSOFF] ID :-[RVSON]"
;UT$:PRINT
263 PRINT"THIS PROCEDURE WILL RE
NAME ALL FILES"
264 PRINT"CONTINUE [RVSON](Y/N)[
RVSOFF]":DO:GETAS:LOOPUNTILAS="Y
"ORAS="N"
265 IFAS="N"THENRETURN
266 FL$="":FORXT=2TOCO:PRINT"[HO
ME][DOWN][DOWN][DOWN][DOWN]CHANG
ING :-";NAS(XT)+""
267 NE$(XT)=LEFT$(NAS(XT),14)+CH
R$(160)+":":
268 RENAME D0,(NAS(XT)) TO (NE$(
XT)),UB
269 IFDSTHENPRINTCHR$(7);CHR$(15
):CHAR1,0,24,DS$,1:SLEEP1:PRINT
"[UP]"CHR$(27)"@";NEXT:ELSE NEXT
270 RETURN
271 :
272 REM **** RENAME ****
273 :
274 GOSUB302
275 PRINT:FORXT=2TOCO
276 PRINT"[HOME][DOWN][DOWN][DOW
N][DOWN][DOWN]";SC$;":-";CHR$(2
7)"Q";NAS(XT);"[RVSON](Y/N/C)[R
VSOFF]"
277 DO:GET AS:LOOP UNTILAS="Y"OR
AS="N"ORAS="C"
278 IFAS="Y"ANDSC=1THENSCLATCH (
NAS(XT)):FL$="":GOTO284
279 IF AS="C"THENPRINTCHR$(15);C
HR$(7);"OK:- CANCELLED":GOTO288
280 IFAS="Y"ANDSC=0 THENBEGIN:QU
$="":PRINT"[HOME][DOWN][DOWN][DOW
N][DOWN][DOWN]";CHR$(27);"@ENTE
R NEW NAME FOR ";NAS(XT):INPUT Q
US
281 IFQU$="":THEN287
282 NE$(XT)=QU$
283 RENAME D0,(NAS(XT)) TO (NE$(
XT)),UB :FL$="":BEND
284 IFDSTHENBEGIN:PRINTCHR$(7);C
HR$(15):CHAR1,0,24,DS$,1
285 IFSC=0THEN CHAR1,0,23,"BLANK
NAME TO MOVE ON OR RE-ENTER NAM
E":SLEEP2
286 SLEEP2:DCLEAR:PRINT"[UP][UP]
[UP]"CHR$(27)"@":BEND:GOTO280
287 NEXT:PRINTDS$:DCLEAR:RETURN
288 SLEEP3:RETURN
289 :
290 REM **** ERROR TRAP ****
291 :
292 FAST:IFREGR(0)=5THENWINDOW0,
0,79,24:ELSEWINDOW0,0,39,24
293 IFER=5ANDDV=4THENDV=3:SLOW:E
LSEIFER=5THENSLOW:PE$=BE$+ERR$(E
R):CHAR1,0,24,PE$,1:SLEEP3
294 FL$="":GOTO29
295 :
296 REM **** READ DATA FOR MENU

```

```

****
297 :
298 FORXT=4TOYU:READ S$(XT)
299 DO:S$(XT)="- "+S$(XT)+" "
300 LOOP UNTIL LEN(S$(XT))>30
301 S$(XT)=LEFT$(S$(XT),30):NEXT
:RETURN
302 :
303 REM ** PRINT DETAILS ON SCRE
EN **
304 :
305 SCNCLR:PRINT"CURRENT DISK :-
[RVSON]";NAS(1);"[RVSOFF] ID :-[
RVSON]";UT$:PRINT
306 IF CO=1THEN:PRINTCHR$(7);CHR
$(15)"NO FILES ON THIS DISK":SLE
EP3:RETURN
307 PRINT "[HOME][DOWN][DOWN]SEL
ECT FILENAMES TO ";SC$
308 RETURN
309 :
310 REM **** LOCK/UNLOCK
****
311 REM **** DELETE/RESTORE FILE
S ****
312 :
313 GOSUB 317:RETURN
314 :
315 REM **** FILETYPE EDITOR
****
316 :
317 SCNCLR
318 OPEN15,8,15,"I":IFDSTHENSCLN
LR:GOTO186
319 OPENS,8,5,"#":PRINT"[CLEAR][
RIGHT][RIGHT][RIGHT][RIGHT][RIGH
T][RIGHT][RIGHT][RIGHT][RIGHT][R
IGHT][RIGHT][RIGHT]FILETYPE EDIT
OR":TR=18:SE=1
320 PRINT#15,"U1";5;0;TR;SE:PT=T
R:PS=SE
321 PRINT#15,"B-P",5;0:GET#5,TR$
,SE$:TR=ASC(TR$):SE=ASC(SE$)
322 B=2:FORN=1TO8:PRINT#15,"B-P"
;5;8
323 GET#5,IS,DT$,DB$
324 T=ASC(IS):DB=ASC(DB$):DT=ASC
(DT$):IFDT=0THEN335:REM LAST NA
ME NO TRACK
325 IFT>0ANDT=128<6THENT=128:L
$="":GOTO327
326 IFT>0ANDT=192<6THENT=192:L
$="LOCKED "
327 PRINT"[HOME][DOWN][DOWN][DOW
N][DOWN][DOWN]"CHR$(27)"@FOUND :
-":PRINT
328 FORI=1TO16
329 GET#5,A$:IFAS="":THENAS=CHR$(
0)
330 PRINTAS;
331 NEXTI:PRINT"[s*][RVSON]";L
$;T$(I);:GOSUB338:B=B+32
332 IFKE$="Q"THEN337
333 NEXT N
334 IFASC(TR$)<>0THEN320:REM NOT
LAST DIRECTORY SECTOR YET
335 CLOSE5:REM CLOSE
336 CLOSE15:REM EVERYTHING
337 RETURN
338 :
339 REM **** GET ELECTION ****
340 :
341 KE$="":PRINT"[HOME][DOWN][DOW
N][DOWN][DOWN][DOWN][DOWN][DOWN]
[DOWN][DOWN][RVSON][RVSOFF]-LO
CK [RVSON][RVSOFF]-UNLOCK [RVSO
N][RVSOFF]-DELETE [RVSON][RVSO

```

```

FF)-RESTORE"
342 PRINT"[RUSON]Q[RUSOFF]-QUIT"
;:INPUT [RUSON]RETURN[RUSOFF] =
NO CHANGE";KES
343 IFKES$=""ORASC(KES$)=0 THENRE
TURN
344 IFKES<>"L"ANDKES<>"U"ANDKES$
>"R"ANDKES<>"D"ANDKES<>"Q"THEN34
1
345 IFKES$="Q"THENCLOSES:CLOSE15:
RETURN
346 IFKES$="L"THENI=ASC(I$)OR64
347 IFKES$="U"THENI=ASC(I$)AND191
348 IFKES$="D"THENI=0
349 IFKES$="R"THENGOSUB359
350 :
351 REM ***WRITE MODFCTION TO DI
SK***
352 :
353 PRINT#15,"B-P";5;B
354 PRINT#5,CHR$(I);
355 PRINT#15,"U2";5;0;PT;PS
356 IFKES$="R"THENBEGIN:PRINT"IT
IS ADVISABLE TO COLLECT"
357 PRINT"THE DISK TO ENSURE A V
ALID BAM":BEND
358 FORXI=0TO500:NEXT:RETURN
359 :
360 REM SELECT TYPE OF FILE
361 :
362 PRINT"[HOME][DOWN][DOWN][DOW
N][DOWN][DOWN]"CHR$(27)"@";:FORX
T=0TO4:PRINTXI;";-";TY$(XT):NEXT
XI
363 PRINT"SELECT FILETYPE TO RES

```

```
TIORE": INPUTX
364 IFX>40RX<0THENPRINT"CUP]":GO
TO363
365 T=X+128:RETURN
366 DATARENAME FILES,INITIALIZE
DRIVE,DISPLAY DIRECTORY,MAKE A B
OOT DISK,PRINT DIRECTORY
367 DATADELETE FILES,FORMAT A DI
SK,COLLECT,LOCK/UNLOCK FILE,DELE
TE/RECOVER FILES,QUIT,SELECT COL
ORS
368 DATAPUT COLON AFTER F/NAME,R
ENAME FILES,GO TO MAIN MENU
369 REM
370 REM SELECT SCREEN COLORS
371 REM
372 SCNCLR:L1$="- [s U][s *][s *]
[s *][s *][s *][s *][s *][s *]
[s *][s *][s *][s *][s *][s *]
[s *][s *][s *][s *][s *][s *]
[s *][s *][s *][s *][s *][s *]
[s *][s *][s *][s *][s *][s *]
[s *][s *]"
373 L2$="- [s J][s *][s *][s *][s *]
[s *][s *][s *][s *][s *][s *]
[s *][s *][s *][s *][s *][s *]
[s *][s *][s *][s *][s *][s *]
[s *][s *][s *][s *][s *][s *]
[s *][s *][s *][s *][s *][s *]
[s *][s *][s *][s *][s *][s *]
[s *][s *]"
374 SCNCLR:PRINTL1$:PRINT " [s -
] COLOR SELECTION MENU
[s -]":PRINTL2$
375 WINDOW0,12,39,23
376 SCNCLR:PRINTL1$:PRINT " [s -
```

```

]
[ s - ]":PRINTL2$:INPUTS:IFS<1
ORS>16THENPRINTERR$(14):GOTO376:
ELSECOLOR4,S
377 GOSUB390
378 IFAS<>"Y"THEN376
379 REM FRGR(0)=5THENCOLOR
6,S:ELSECOLOR0,S
380 SCNCLR:PRINTL1$:PRINT " [ s -
] SELECT BORDER COLOUR
[ s - ]":PRINTL2$:INPUTB:IFB<1
ORB>16THENPRINTERR$(14):GOTO380:
ELSECOLOR4,B
381 GOSUB390
382 IFAS<>"Y"THEN380
383 REM COLOR4,B
384 SCNCLR:PRINTL1$:PRINT " [ s -
] SELECT PRINT COLOUR.
[ s - ]":PRINTL2$:INPUTPP:IFPP
<10RPP>16THENPRINTERR$(14):GOTO3
84:ELSECOLOR4,PP
385 GOSUB390
386 IFAS<>"Y"THEN384
387 REM COLORS,S
388 COLOR4,B:COLORS,PP:COLOR (RG
R(0)),S:IFRGR(0)=5THENWINDOW0,0,
79,24:RETURN:ELSEWINDOW0,0,39,24
:RETURN
389 AS="":INPUTAS:IFAS=" "THEN389
:ELSE RETURN
390 SCNCLR:PRINTL1$:PRINT " [ s -
] IS YOUR SELECTION OK?
[ s - ]":PRINTL2$
391 GOSUB389
392 RETURN

```

LIFESAVERS	C64	BASIC PROTECTOR	1/1
<p>These short Basic and machine code routines will enable a Basic program to be protected against prying eyes trying to LIST it.</p> <p>They work by changing the LIST vector to point to a place in memory where a message is stored. The routine is situated at 53210 (\$CFDA) and is called by SYS 53210.</p> <p>To use the program to its full effectiveness, the routine should be incorporated in to a short auto-run loader program.</p> <p>P.A.Eves</p> <pre> 1 DATA 169,229,160,207,141,6,3,1 40,7,3,96,169,239,160,207,32,30, 171,76,116 2 DATA 164,147,73,76,69,71,65,76 ,32,69,78,84,82,89,46,46,46,00 3 FORA=53210TO53247:READB:POKEA, B:NEXT 4 SYS53210 10 ; program start address </pre>	<pre> 20      *=\$CFDA 30 ; 10 byte of new pointer 40      LDA #&lt;NEWPNT 50 ; hi byte of new pointer 60      LDY #&gt;NEWPNT 70 ; put into 10 byte of LIST vector 80      STA \$0306 90 ; put into hi byte of LIST vector 100     STY \$0307 110     RTS 120 ; 10 byte of start of message 130 NEWPNT LDA #&lt;TEXT 140 ; hi byte of start of message 150     LDY #&gt;TEXT 160 ; print the message 170     JSR \$A81E 180 ; back to Basic 190     JMP \$A474 120 TEXT .BYT \$23,\$49,\$4C,\$4C ,\$45,\$47,\$41,\$4C,\$20,\$45,\$4E,\$54 ,\$52,\$59,\$2E 130     .BYT \$2E,\$2E,\$00 140     .END </pre>		



# New Characters on the MPS 801/3

*Fed up with your MPS 801/3 character set? Now you can design your own characters.*

**W**hen I bought my MPS 803 printer, I was very disappointed with the print quality; no true descenders and no chance of having Near Letter Quality (NLQ).

To overcome the most annoying problem of the two, no true descenders, I could think of two ways:

One, to print each line of text with two passes of the print head, the first to print the top part of the letters, the second to print the descenders.

Two, to re-define the character set so it's a little more squashed but has true descenders.

I chose the latter, partly because it would be quicker to print the text, and partly because I could design lots of different character sets more easily and they would take up less memory.

Presented here is that program, with a few extra frills. It is not very difficult to use, is as fast as it can be, because the printer is operating in bit image mode, and also uses up very little memory.

The character sets take up \$0300 (768) bytes and can be stored anywhere in memory (except from \$0000-\$0800 (2048), obviously!), including under the ROMs.

## Getting it in

The program is presented here as a Basic listing called *PRINTER DRIVER*. Type this in using our *SYNTAX CHECKER* program that can be found with the *LISTINGS* article. Make sure that you SAVE the program to tape or disk before you RUN it.

If the program finds any errors when

you RUN it it will indicate the line where the error has occurred. When you've got everything going O.K., type:

SYS 52603

to initialise it, or

SYS 52600

to switch it off, which I doubt will be necessary. If you use RUN STOP/RESTORE, you'll have to re-initialise it. Nothing should appear to have happened after initialisation. You can now use the printer as normal, but everything will be printed through the driver.

There are a few more things that you may need to know:

**OPEN X,Y,7** — will print in UPPER CASE/lower case, as normal.

**OPEN X,Y** — will print in UPPER CASE only (shifted letters will be printed in lower case.)

**CHRS(27)CHRS(SET)** — selects the character set, where set is the most significant byte of the start address of the set i.e. If the start address of the character set is \$FC00 (64512), the most significant byte is  $64512/256=252$ .

**CHRS(18)** — will print in inverse characters until you de-select it with CHRS(146)

**CHRS(8)** — will print in Bit Image mode until you de-select it with CHRS(15)

**CHRS(14)** — will print in enhanced characters until you de-select it with CHRS(15)

All of the above CHRS codes must be printed after a PRINT(No.)X instruction, where X is the logical file number (see printer manual), and Y is the device number (normally 4).

## Printing Rubbish

If you have typed in the Driver and initialised it, don't be surprised when you get a whole load of garbage printed out. This is because you have to do more work; you have to type in a character set. There are five sets that I have designed included here, choose which you think will be the most useful, and type it in, or better still, type them all in. Again use the *SYNTAX CHECKER* to check your typing. The default character set that the driver software will print starts at 64512. If you wish to print from another set use:

PRINT(No.)X,CHRS(27)CHRS(start address/256)

The character sets can be relocated to any position in memory, (I have positioned them under the KERNAL ROM) provided that the start address divided by 256 gives a whole number.

## On your own

If you plan to design your own character sets, you'll find the following information useful:

When designing the characters, each number in the list of data stands for one

By Mathew Bittleston

row which is printed. The format in which the information is stored is shown in Figure 1.

The important thing about the above is that it is different from when you define normal U.D.G.s (for games etc.). For the printer, the rows are VERTICAL instead of horizontal.

The MPS 803 can only print 7 dots vertically, so in each byte there is one bit not used. I've used this bit to tell the driver whether to print that row or not; if bit 7 is SET, then that row will be printed, if it's NOT, then it won't be. This makes it possible to define character sets of different widths, or even a proportional set. The maximum width of a character is 8 rows, the normal MPS803 character width is just 6 rows.

Now let me try to clarify what I've just said: Look at Figure 1, from that you can see that because bit 7 is SET, row 0 will be printed. In fact, if you look carefully at it, you can see that nothing will be printed, but the print head will move one row to the right, so that all the letters aren't joined together. Now look at row 7, bit 7 is NOT set, so that line will not be sent to the printer.

The line of data for the letter 'M' I have designed would be:

DATA 128,191,130,140,191,0,0

## NLQ?

I have thought of one way of producing a form of NLQ (???) on this printer:

First, print out your text, remembering **exactly** how far up and across the paper was.

Now, put the paper back into the same position as last time, and print the text again. If you do this just slightly wrong, you'll get double images of every letter, extremely annoying, but, do it right, and you have a kind of NLQ on the MPS 803.

## Important Note

This program should work with most Commodore printers that will print in Bit Image Mode, e.g. the MPS 801 but we haven't been able to test it with them all. The program is not designed to work together with commercial software. If you want to try it go ahead, but, we can't guarantee what the results will be.

Figure 1

Bit	Value	Row: 0 1 2 3 4 5 6 7
0.....	1	..... O * O O O * O O This
1.....	2	..... O * * O * * O O example
2.....	4	..... O * O * O * O O shows
3.....	8	..... O * O * O * O O 'M'
4.....	16	..... O * O O O * O O from
5.....	32	..... O * O O O * O O the
6.....	64	..... O O O O O O O O Descender
7.....	128	..... * * * * * O O set.

128 130 130 0  
191 140 191 0

### PROGRAM: DESCENDER LDR

```

AE 100 REM * DRIVER CANNOT BE R OF 1030 DATA 0,141,153,206,141,
    ELOCATED * 152,206,76,128,205,201,18,16
88 110 SA=52500 27
A8 120 B=0:FORX=0TO11:READA:B=B E8 1032 DATA 208,6,141,150,206,
    +A:POKESA+X,A:NEXT:SA=SA+X:R 76,128,205,201,146,208,8,168
    EADA:IFA=BTHEN120 3
06 130 IFA>0THENPRINT"DATA ERRO 46 1034 DATA 169,0,141,150,206,
    R IN LINE"PEEK(63)+PEEK(64)* 76,128,205,201,16,208,6,1506
    256
52 1000 DATA 76,51,205,169,0,14 02 1036 DATA 141,155,206,76,128
    1,150,206,141,151,206,141,16 ,205,96,201,32,144,24,201,16
    37 09
43 1002 DATA 152,206,141,153,20 12 1038 DATA 96,176,6,56,233,32
    6,141,155,206,160,205,169,64 ,76,24,206,201,192,144,1442
    ,1958 C7 1040 DATA 10,201,224,176,6,5
7D 1004 DATA 140,39,3,141,38,3, 6,233,128,76,24,206,169,1509
    96,172,157,206,173,156,1324
65 1006 DATA 206,140,39,3,141,3 C1 1042 DATA 15,32,168,255,104,
    8,3,96,234,141,148,206,1395 76,168,255,133,251,165,185,1
5F 1008 DATA 165,154,201,3,208, 807
    6,173,148,206,76,22,231,1593 B8 1044 DATA 201,255,208,10,165
    ,251,201,32,144,4,73,96,1640
17 1010 DATA 176,3,76,219,241,1 BC 1046 DATA 133,251,169,0,133,
    73,148,206,72,32,95,205,1646 252,6,251,38,252,6,251,1742
15 1012 DATA 76,243,205,173,152 B4 1048 DATA 38,252,6,251,38,25
    ,206,208,33,173,155,206,208, 2,173,154,206,24,101,252,174
    2038 7
FB 1014 DATA 46,173,148,206,201 18 1050 DATA 133,252,140,149,20
    ,27,240,13,173,151,206,240,1 6,142,148,206,169,8,32,168,1
    824 753
72 1016 DATA 55,173,148,206,141 99 1052 DATA 255,160,7,120,32,1
    ,154,206,169,0,141,151,206,1 38,206,177,251,153,159,206,1
    750 864
CE 1018 DATA 104,104,104,24,96, A5 1054 DATA 136,16,248,32,138,
    173,148,206,201,15,208,5,138 206,88,200,185,159,206,16,16
    8 30
C4 1020 DATA 169,0,141,152,206, 32 1056 DATA 18,174,150,206,240
    104,104,104,76,168,255,173,1 ,2,73,127,174,153,206,240,17
    652 63
91 1022 DATA 148,206,141,155,20 B3 1058 DATA 3,32,168,255,32,16
    6,240,225,169,32,32,243,205, 8,255,200,192,8,208,228,1749
    2002
DA 1024 DATA 206,155,206,208,24 AA 1060 DATA 172,149,206,174,14
    6,76,128,205,173,148,206,201 8,206,173,155,206,208,2,104,
    ,2158 1903
4E 1026 DATA 8,208,6,141,152,20 83 1062 DATA 24,96,165,1,73,7,1
    6,76,145,205,201,14,208,1570 33,1,96,0,0,0,596
84 1028 DATA 6,141,153,206,76,1 DF 1064 DATA 0,0,0,0,0,0,252,0,
    28,205,201,15,208,11,169,151 202,241,0,0,695
    9 E1 1066 DATA 0,0,0,0,0,0,0,0,0,0,
    0,0,0,-1

```



## PROGRAM: DESCENDER SET

```

70 100 REM * DESCENDER *
8C 110 SA=64512
42 120 PRINT"ESC CODE FOR SET -
    CHR$(27)CHR$("SA/256")"
23 130 B=0:FORX=0TO11:READA:B=B
    +A:POKESA+X,A:NEXT:SA=SA+X:R
    EADA:IFA=BTHEN130
04 140 IFA>0THENPRINT"DATA ERRO
    R IN LINE"PEEK(63)+PEEK(64)*
    256
58 1000 DATA 128,128,128,128,12
    8,128,0,0,128,128,128,222,13
    74
90 1002 DATA 128,128,0,0,128,13
    2,131,128,132,131,0,0,1038
40 1004 DATA 128,148,190,148,19
    0,148,0,0,128,132,170,255,16
    37
CD 1006 DATA 170,144,0,0,128,17
    7,137,164,163,128,0,0,1211
D3 1008 DATA 128,148,170,255,17
    0,162,0,0,128,128,128,132,15
    49
DC 1010 DATA 131,128,0,0,128,15
    6,162,193,128,128,0,0,1154
54 1012 DATA 128,128,193,162,15
    6,128,0,0,128,170,156,190,15
    39
E1 1014 DATA 156,170,0,0,128,13
    6,136,190,136,136,0,0,1188
77 1016 DATA 128,128,128,192,17
    6,128,0,0,128,136,136,136,14
    16
E1 1018 DATA 136,136,0,0,128,12
    8,128,176,176,128,0,0,1136
99 1020 DATA 128,160,144,136,13
    2,130,0,0,128,158,177,173,14
    66
32 1022 DATA 163,158,0,0,128,12
    8,162,191,160,128,0,0,1218
19 1024 DATA 128,178,169,169,16
    9,166,0,0,128,145,161,161,15
    74
40 1026 DATA 165,155,0,0,128,13
    6,140,138,191,136,0,0,1189
7C 1028 DATA 128,151,165,165,16
    5,153,0,0,128,156,166,165,15
    42
C3 1030 DATA 165,152,0,0,128,12
    9,129,185,133,131,0,0,1152
89 1032 DATA 128,154,165,165,16
    5,154,0,0,128,134,169,169,15
    31
FF 1034 DATA 153,142,0,0,128,12
    8,128,164,164,128,0,0,1135
38 1036 DATA 128,128,128,196,18
    0,128,0,0,128,136,148,162,14
    62
CB 1038 DATA 162,128,0,0,128,14
    8,148,148,148,148,0,0,1158
10 1040 DATA 128,162,162,148,13
    6,128,0,0,128,130,129,169,14
    20
96 1042 DATA 133,130,0,0,128,15
    8,161,173,169,134,0,0,1186
2D 1044 DATA 128,144,170,170,17
    0,188,0,0,128,191,164,162,16
    15
13 1046 DATA 162,156,0,0,128,15
    6,162,162,162,162,0,0,1250
AE 1048 DATA 128,156,162,162,16
    4,191,0,0,128,156,170,170,15

```

```

B7
E6 1050 DATA 170,140,0,0,128,12
    8,132,190,133,129,0,0,1150
D2 1052 DATA 128,140,210,210,21
    0,190,0,0,128,191,132,130,16
    69
00 1054 DATA 130,188,0,0,128,12
    8,164,189,160,128,0,0,1215
26 1056 DATA 128,128,192,197,18
    9,128,0,0,128,191,136,140,15
    57
1B 1058 DATA 146,160,0,0,128,12
    8,161,191,160,128,0,0,1202
46 1060 DATA 128,190,130,188,13
    0,188,0,0,128,190,132,130,15
    34
3D 1062 DATA 130,188,0,0,128,15
    6,162,162,162,156,0,0,1244
45 1064 DATA 128,254,146,162,16
    2,156,0,0,128,156,162,162,16
    16
B1 1066 DATA 146,254,0,0,128,19
    0,132,130,130,132,0,0,1242
80 1068 DATA 128,164,170,170,17
    0,144,0,0,128,130,159,162,15
    25
A0 1070 DATA 162,144,0,0,128,15
    8,160,160,160,190,0,0,1262
A7 1072 DATA 128,142,144,160,14
    4,142,0,0,128,158,160,152,14
    58
07 1074 DATA 160,158,0,0,128,16
    2,182,136,182,162,0,0,1270
E9 1076 DATA 128,142,208,208,20
    8,190,0,0,128,162,178,170,17
    22
68 1078 DATA 166,162,0,0,128,12
    8,255,193,193,128,0,0,1353
75 1080 DATA 128,168,190,169,16
    9,162,0,0,128,128,193,193,16
    28
42 1082 DATA 255,128,0,0,128,12
    8,130,191,130,128,0,0,1218
AS 1084 DATA 128,136,156,136,13
    6,136,0,0,136,136,136,136,13
    72
BE 1086 DATA 136,136,0,0,128,18
    8,138,137,138,188,0,0,1189
FB 1088 DATA 128,161,191,165,16
    5,154,0,0,128,158,161,161,15
    72
2C 1090 DATA 161,146,0,0,128,16
    1,191,161,146,140,0,0,1234
9F 1092 DATA 128,161,191,165,16
    5,161,0,0,128,161,191,165,16
    16
3E 1094 DATA 133,129,0,0,128,15
    8,161,161,169,185,0,0,1224
B2 1096 DATA 128,191,132,132,13
    2,191,0,0,128,128,161,191,15
    14
9D 1098 DATA 161,128,0,0,128,14
    4,160,161,159,129,0,0,1170
6E 1100 DATA 128,128,191,140,14
    6,161,0,0,128,161,191,161,15
    35
59 1102 DATA 160,176,0,0,128,19
    1,130,140,130,191,0,0,1246
2C 1104 DATA 128,191,130,132,13
    6,191,0,0,128,158,161,161,15
    16
03 1106 DATA 161,158,0,0,128,16
    1,191,169,137,134,0,0,1239
4A 1108 DATA 128,158,161,169,14
    5,174,0,0,128,161,191,137,15
    52

```

```

30 1110 DATA 153,166,0,0,128,14
    6,165,165,165,152,0,0,1240
9C 1112 DATA 128,131,161,191,16
    1,131,0,0,128,159,160,160,15
    10
24 1114 DATA 160,159,0,0,128,13
    5,152,160,152,135,0,0,1181
D2 1116 DATA 128,159,160,156,16
    0,159,0,0,128,161,179,140,15
    30
48 1118 DATA 179,161,0,0,128,13
    1,164,188,164,131,0,0,1246
92 1120 DATA 128,163,177,173,16
    3,177,0,0,136,136,136,255,16
    44
29 1122 DATA 136,136,0,0,0,85,4
    2,85,0,0,0,0,0,484
84 1124 DATA 0,0,0,0,127,0,0,0,
    0,4,60,4,195
88 1126 DATA 28,2,0,0,0,62,65,9
    3,85,65,62,0,-1

```

## PROGRAM: MODERN SET

```

CE 100 REM * MODERN *
90 110 SA=63744
42 120 PRINT"ESC CODE FOR SET -
    CHR$(27)CHR$("SA/256")"
23 130 B=0:FORX=0TO11:READA:B=B
    +A:POKESA+X,A:NEXT:SA=SA+X:R
    EADA:IFA=BTHEN130
04 140 IFA>0THENPRINT"DATA ERRO
    R IN LINE"PEEK(63)+PEEK(64)*
    256
B1 1000 DATA 128,128,128,128,12
    8,128,128,0,128,128,172,175,
    1499
44 1002 DATA 128,128,128,0,128,
    134,135,128,134,135,128,0,13
    06
06 1004 DATA 128,138,186,191,13
    8,191,138,0,128,174,170,255,
    1837
55 1006 DATA 170,186,186,0,128,
    179,187,152,132,180,179,0,16
    79
86 1008 DATA 128,188,191,165,17
    3,144,168,0,128,128,128,133,
    1674
66 1010 DATA 131,128,128,0,128,
    188,250,225,193,128,128,0,16
    27
C5 1012 DATA 128,128,193,225,25
    0,188,128,0,128,170,156,190,
    1884
D3 1014 DATA 156,170,128,0,128,
    136,136,184,190,136,136,0,15
    00
60 1016 DATA 128,128,128,208,17
    6,128,128,0,128,136,136,136,
    1560
19 1018 DATA 136,136,128,0,128,
    128,128,176,176,128,128,0,13
    92
3C 1020 DATA 128,160,176,152,14
    0,130,129,0,128,184,191,169,
    1687
27 1022 DATA 165,163,191,0,128,
    128,184,191,128,128,128,0,15
    34
81 1024 DATA 128,184,185,169,16
    9,173,175,0,128,160,169,169,

```

## PROGRAM: ITALIC SET

```

1809
72 1026 DATA 169,191,184,0,128,
143,140,136,184,190,136,0,16
01
35 1028 DATA 128,175,173,169,16
9,185,184,0,128,184,191,169,
1855
AF 1030 DATA 168,168,184,0,128,
129,129,185,185,133,131,0,15
40
D2 1032 DATA 128,184,191,169,16
9,173,191,0,128,143,139,137,
1752
15 1034 DATA 185,191,128,0,128,
128,128,148,148,128,128,0,14
40
90 1036 DATA 128,128,128,212,18
0,128,128,0,128,128,152,180,
1620
1F 1038 DATA 226,193,128,0,128,
148,148,148,148,148,128,0,15
43
D4 1040 DATA 128,128,193,226,18
0,152,128,0,128,131,129,217,
1740
8C 1042 DATA 217,139,143,0,129,
158,161,173,173,169,142,0,16
04
62 1044 DATA 128,184,191,137,13
7,139,191,0,128,191,185,169,
1780
05 1046 DATA 171,175,184,0,128,
184,191,161,161,163,179,0,16
97
72 1048 DATA 128,184,191,161,16
2,164,184,0,128,184,191,171,
1848
6F 1050 DATA 169,169,160,0,128,
184,191,139,137,137,128,0,15
42
F9 1052 DATA 128,184,188,162,16
1,161,185,0,128,184,191,136,
1808
9C 1054 DATA 136,140,191,0,128,
128,184,191,131,128,128,0,14
85
A0 1056 DATA 128,184,184,160,19
1,131,128,0,128,184,191,136,
1745
C9 1058 DATA 140,154,185,0,128,
184,191,160,160,176,176,0,16
54
99 1060 DATA 128,184,191,129,14
3,129,191,0,128,184,191,129,
1727
D7 1062 DATA 129,131,191,0,128,
184,191,161,161,163,191,0,16
30
FD 1064 DATA 128,184,191,137,13
7,139,143,0,128,184,191,161,
1723
E5 1066 DATA 169,147,175,0,128,
184,191,137,153,171,143,0,15
98
BB 1068 DATA 128,144,166,169,16
9,187,187,0,128,129,185,191,
1783
B7 1070 DATA 129,131,131,0,128,
184,191,160,160,163,191,0,15
68
A1 1072 DATA 128,152,191,160,16
0,147,143,0,128,184,191,160,
1744
FC 1074 DATA 184,163,191,0,128,
176,187,140,140,139,176,0,16
24

```

```

B5 1076 DATA 128,143,139,184,18
4,140,143,0,128,160,177,185,
1711
FD 1078 DATA 169,165,163,0,128,
128,248,255,193,128,128,0,17
05
B3 1080 DATA 128,160,184,191,16
9,171,163,0,128,128,192,249,
1863
41 1082 DATA 255,128,128,0,128,
132,242,255,130,132,128,0,16
58
80 1084 DATA 136,136,136,168,18
6,156,136,0,0,0,0,255,1309
D9 1086 DATA 255,0,0,255,128,18
4,191,137,137,139,191,0,1617
9D 1088 DATA 128,191,185,169,17
1,175,184,0,128,184,191,161,
1867
1B 1090 DATA 161,163,179,0,128,
184,191,161,162,164,184,0,16
77
18 1092 DATA 128,184,191,171,16
9,169,160,0,128,184,191,139,
1814
4C 1094 DATA 137,137,128,0,128,
184,188,162,161,161,185,0,15
71
0A 1096 DATA 128,184,191,136,13
6,140,191,0,128,128,184,191,
1737
B0 1098 DATA 131,128,128,0,128,
184,184,160,191,131,128,0,14
93
F6 1100 DATA 128,184,191,136,14
0,154,185,0,128,184,191,160,
1781
B6 1102 DATA 160,176,176,0,128,
184,191,129,143,129,191,0,16
07
E4 1104 DATA 128,184,191,129,12
9,131,191,0,128,184,191,161,
1747
79 1106 DATA 161,163,191,0,128,
184,191,137,137,139,143,0,15
74
AD 1108 DATA 128,184,191,161,16
9,147,175,0,128,184,191,137,
1795
A7 1110 DATA 153,171,143,0,128,
144,166,169,169,187,187,0,16
17
A9 1112 DATA 128,129,185,191,12
9,131,131,0,128,184,191,160,
1687
01 1114 DATA 160,163,191,0,128,
152,191,160,160,147,143,0,15
95
EB 1116 DATA 128,184,191,160,18
4,163,191,0,128,176,187,140,
1832
87 1118 DATA 140,139,176,0,128,
143,139,184,184,140,143,0,15
16
C6 1120 DATA 128,160,177,185,16
9,165,163,0,128,128,248,255,
1906
1C 1122 DATA 193,128,128,0,128,
160,184,191,169,171,163,0,16
15
7E 1124 DATA 128,128,192,249,25
5,128,128,0,128,132,242,255,
1965
E6 1126 DATA 130,132,128,0,136,
136,136,168,186,156,136,0,-1

```

```

6D 100 REM * ITALIC *
94 110 SA=62976
42 120 PRINT"ESC CODE FOR SET =
CHR$(27)CHR$("SA/256")"
23 130 B=0:FORX=0TO11:READA:B=B
+A:POKESA+X,A:NEXT:SA=SA+X:R
EADA:IFA=BTHEN130
04 140 IFA>0THENPRINT"DATA ERRO
R IN LINE"PEEK(63)+PEEK(64)*
256
5D 1000 DATA 128,128,128,128,12
8,128,0,0,128,128,192,144,13
60
E1 1002 DATA 140,131,0,0,128,13
3,131,128,133,131,0,0,1055
CC 1004 DATA 144,244,156,247,15
6,151,0,0,128,128,164,234,17
52
04 1006 DATA 171,146,0,0,128,17
6,137,173,164,131,0,0,1226
6A 1008 DATA 128,144,172,234,17
1,130,0,0,128,128,128,133,14
96
3C 1010 DATA 131,128,0,0,128,12
8,152,164,194,129,0,0,1154
A5 1012 DATA 128,128,192,161,14
6,140,0,0,128,170,156,190,15
39
E1 1014 DATA 156,170,0,0,128,13
6,136,190,136,136,0,0,1188
85 1016 DATA 128,128,128,208,17
6,128,0,0,128,136,136,136,14
32
E1 1018 DATA 136,136,0,0,128,12
8,128,176,176,128,0,0,1136
88 1020 DATA 128,160,144,136,13
2,130,0,0,128,156,170,161,14
45
B0 1022 DATA 149,142,0,0,128,12
8,160,186,167,128,0,0,1188
52 1024 DATA 128,176,170,169,16
9,134,0,0,128,144,161,165,15
44
05 1026 DATA 155,129,0,0,128,13
6,140,170,157,139,0,0,1154
1C 1028 DATA 128,144,166,165,16
5,153,0,0,128,156,166,165,15
36
0C 1030 DATA 165,153,0,0,128,16
0,145,137,133,131,0,0,1152
D8 1032 DATA 128,144,170,165,16
5,154,0,0,128,166,169,169,15
58
A3 1034 DATA 153,142,0,0,128,12
8,144,148,132,128,0,0,1103
5E 1036 DATA 128,128,208,180,13
2,128,0,0,128,136,148,162,14
78
51 1038 DATA 161,129,0,0,128,14
4,148,148,148,132,0,0,1138
86 1040 DATA 128,160,161,145,13
8,132,0,0,128,128,194,153,14
67
A2 1042 DATA 137,134,0,0,128,15
8,161,173,173,169,142,0,1375
F5 1044 DATA 128,184,164,162,14
6,174,0,0,128,176,174,165,16
01
A3 1046 DATA 164,152,0,0,128,15
2,164,162,162,148,0,0,1232

```



9C	1048 DATA 128,156,162,162,15 6,163,0,0,128,152,172,170,15 49	B2	1108 DATA 128,156,162,169,14 5,174,0,0,128,184,143,153,15 42	54	1,1924 1022 DATA 197,255,190,128,12 8,136,196,254,191,160,128,12 8,2091
55	1050 DATA 170,132,0,0,128,19 2,200,190,137,129,0,0,1278	CE	1110 DATA 169,134,0,0,128,14 4,162,165,165,153,0,0,1220	D9	1024 DATA 128,194,227,245,23 3,239,198,128,128,162,227,20 1,2310
9A	1052 DATA 128,204,210,210,17 0,158,0,0,128,160,158,133,16 59	40	1112 DATA 128,129,185,135,12 9,129,0,0,128,152,167,160,14 42	7A	1026 DATA 201,237,182,128,14 4,152,148,146,255,255,176,12 8,2152
15	1054 DATA 164,152,0,0,128,12 8,180,172,165,129,0,0,1218	9F	1114 DATA 184,135,0,0,128,15 2,167,160,144,143,0,0,1213	2C	1028 DATA 128,167,213,197,20 5,221,184,128,128,190,255,20 9,2225
05	1056 DATA 128,176,192,180,14 1,129,0,0,128,160,158,137,15 29	C0	1116 DATA 128,158,161,152,16 0,159,0,0,128,160,147,140,14 93	EB	1030 DATA 201,251,176,128,12 8,130,131,243,251,167,131,12 8,2065
D7	1058 DATA 180,162,0,0,128,12 8,184,166,165,131,0,0,1244	D6	1118 DATA 140,179,128,0,128, 182,201,200,190,129,0,0,1477	06	1032 DATA 128,182,237,201,20 1,219,182,128,128,150,175,20 1,2132
1C	1060 DATA 128,188,130,140,16 2,156,0,0,128,176,142,132,14 82	66	1120 DATA 128,162,177,173,16 3,145,0,0,136,136,136,255,16 11	AA	1034 DATA 229,191,158,128,12 8,128,160,180,150,130,128,12 8,1838
4D	1062 DATA 178,140,0,0,128,18 4,164,162,146,140,0,0,1242	84	1122 DATA 136,136,0,0,0,0,0, 0,0,0,0,0,272	54	1036 DATA 128,128,192,180,18 2,130,128,128,128,136,156,18 2,1798
C8	1064 DATA 128,248,150,162,16 2,156,0,0,0,136,148,210,1500	8B	1124 DATA 0,0,0,0,0,0,0,0,0, 0,0,0,0	0F	1038 DATA 227,193,193,128,12 8,148,148,148,148,148,148,12 8,1885
A6	1066 DATA 178,158,128,0,128, 162,158,132,130,132,0,0,1306	C5	1126 DATA 0,0,0,0,0,0,0,0,0, 0,0,0,-1	C4	1040 DATA 128,193,193,227,18 2,156,136,128,128,136,134,16 3,1904
58	1068 DATA 128,160,164,170,17 0,146,0,0,128,128,180,174,15 48	4F	1128 DATA 128,128	D1	1042 DATA 235,143,134,128,12 8,190,193,221,221,209,174,12 8,2104
4F	1070 DATA 165,144,0,0,128,18 4,166,160,152,166,0,0,1265	PROGRAM: GOTHIC SET		45	1044 DATA 128,148,186,170,17 0,190,188,144,128,130,191,19 1,1964
4D	1072 DATA 128,152,166,160,14 4,142,0,0,128,188,146,184,15 38			53	1046 DATA 164,162,158,156,12 8,160,156,190,162,166,162,14 4,1908
53	1074 DATA 144,142,0,0,128,16 0,150,136,180,130,0,0,1170	8D	100 REM * GOTHIC *	8F	1048 DATA 128,160,156,190,16 2,165,191,191,128,156,190,17 8,1995
9A	1076 DATA 128,204,210,200,18 8,130,0,0,128,160,178,170,16 96	83	110 SA=61440	85	1050 DATA 170,238,180,128,12 8,136,196,190,191,149,133,12 9,1968
65	1078 DATA 166,146,0,0,128,22 4,220,195,129,129,0,0,1337	42	120 PRINT"ESC CODE FOR SET = CHR\$(27)CHR\$("SA/256")"	82	1052 DATA 128,172,222,210,20 2,254,188,132,128,130,191,15 9,2116
78	1080 DATA 128,200,248,206,20 1,192,0,0,128,192,192,225,19 12	23	130 B=0:FORX=0TO11:READA:B=B +A:POKESA+X,A:NEXT:SA=SA+X:R EADA:IFA-BTHEN130	AA	1054 DATA 132,196,252,184,12 8,144,168,189,187,160,144,13 6,2020
64	1082 DATA 157,131,0,0,128,12 8,228,154,135,130,4,0,1195	04	140 IFA>0THENPRINT"DATA ERRO R IN LINE"PEEK(63)+PEEK(64)* 256	67	1056 DATA 128,160,192,200,25 3,185,144,128,128,130,191,19 1,2030
47	1084 DATA 136,136,136,168,15 4,140,0,0,136,136,136,136,14 14	39	1000 DATA 128,128,128,128,12 8,128,128,128,128,128,128,22 2,1630	2E	1058 DATA 168,156,180,144,12 8,192,162,190,191,144,128,12 8,1911
1A	1086 DATA 136,136,0,0,128,15 6,162,161,157,163,0,0,1199	37	1002 DATA 239,128,128,128,12 8,138,135,130,128,138,135,13 0,1685	68	1060 DATA 128,190,158,132,14 0,166,190,156,128,190,158,13 2,1868
3E	1088 DATA 128,184,167,165,16 5,154,0,0,128,156,162,161,15 70	D1	1004 DATA 148,254,190,148,21 2,190,190,148,128,164,174,23 5,2181	8D	1062 DATA 194,190,188,144,12 8,220,190,178,171,190,158,12 9,2080
47	1090 DATA 161,146,0,0,128,18 4,167,161,145,142,0,0,1234	A1	1006 DATA 235,186,146,128,12 8,227,179,152,140,230,227,12 8,2106	D2	1064 DATA 128,132,254,254,14 6,162,158,140,128,156,158,16 2,1978
78	1092 DATA 128,184,167,165,16 5,129,0,0,128,184,135,133,15 18	2C	1008 DATA 128,146,183,237,15 7,178,168,236,128,128,128,12 8,1945	22	1066 DATA 146,254,254,160,12 8,132,190,190,148,130,134,13 2,1998
5D	1094 DATA 133,129,0,0,128,15 6,162,225,169,154,0,0,1256	1E	1010 DATA 138,135,130,128,12 8,128,156,190,227,193,128,12 8,1809	92	1068 DATA 128,196,174,186,17 4,186,145,128,128,140,132,15 8,1875
80	1096 DATA 128,184,135,132,18 8,135,0,0,128,128,160,185,15 03	C5	1012 DATA 128,128,193,227,19 0,156,128,128,136,170,190,15 6,1930	F8	1070 DATA 191,164,150,128,12 8,156,190,160,144,190,190,14 4,1935
2C	1098 DATA 167,129,0,0,128,15 2,160,161,153,135,0,0,1185	AE	1014 DATA 156,190,170,136,12 8,136,136,190,190,136,136,12 8,1832		
9C	1100 DATA 128,184,143,148,16 2,129,0,0,128,184,166,161,15 33	67	1016 DATA 128,128,240,224,19 2,128,128,128,128,144,136,14 0,1844		
11	1102 DATA 160,144,0,0,128,19 0,129,142,161,158,0,0,1212	A3	1018 DATA 136,136,136,136,12 8,128,128,176,176,192,128,12 8,1728		
7A	1104 DATA 128,184,135,129,18 5,134,0,0,128,156,162,161,15 02	54	1020 DATA 128,160,176,152,14 0,134,131,129,128,190,255,20		
57	1106 DATA 145,142,0,0,128,18 4,143,137,137,134,0,0,1150				

```

78 1072 DATA 128,132,142,158,17
6,180,158,142,128,156,158,17
6,1834
E0 1074 DATA 156,176,158,156,14
4,162,182,156,156,182,162,13
2,1922
44 1076 DATA 128,132,174,222,20
8,244,190,158,128,129,202,20
2,2117
C3 1078 DATA 190,186,144,128,12
8,128,255,255,193,193,128,12
8,2056
17 1080 DATA 128,208,232,254,22
1,201,202,160,128,128,193,19
3,2248
75 1082 DATA 255,255,128,128,12
8,132,134,255,255,134,132,12
8,2064
69 1084 DATA 128,140,158,191,14
0,140,140,140,152,152,152,15
2,1785
1C 1086 DATA 152,152,152,152,12
8,192,188,158,145,255,190,16
0,2024
FC 1088 DATA 128,226,191,173,16
5,159,138,128,128,156,190,17
7,1959
36 1090 DATA 255,161,146,128,12
8,192,161,191,163,163,191,15
8,2037
08 1092 DATA 128,156,190,229,25
5,229,161,128,128,192,209,19
0,2195
7A 1094 DATA 191,137,137,130,12
8,188,250,195,211,250,178,12
8,2123
E6 1096 DATA 128,194,190,191,13
3,194,190,156,128,192,160,19
0,2046
36 1098 DATA 191,146,128,128,12
8,160,208,193,255,190,130,12
8,1985
E4 1100 DATA 128,194,255,216,18
2,226,224,128,128,192,252,19
0,2315
1C 1102 DATA 227,224,192,128,25
2,190,131,190,156,195,190,15
6,2231
D1 1104 DATA 194,191,191,130,19
5,191,190,128,128,192,190,19
1,2111
B5 1106 DATA 201,197,190,159,12
8,196,191,191,145,143,142,12
8,2011
88 1108 DATA 128,158,191,169,16
5,255,222,192,192,188,190,16
9,2219
4F 1110 DATA 157,183,226,128,19
2,182,190,205,205,190,153,12
8,2139
FC 1112 DATA 128,178,139,207,19
1,191,147,129,128,188,254,19
3,2073
50 1114 DATA 196,254,191,128,12
8,132,158,191,224,226,190,15
9,2177
8A 1116 DATA 128,158,191,196,19
0,225,252,191,128,192,227,21
5,2293
03 1118 DATA 141,220,243,227,12
8,129,191,254,208,217,206,16
6,2330
BF 1120 DATA 128,160,210,202,23
7,191,187,144,140,140,140,25
5,2134
34 1122 DATA 255,140,140,140,14

```

```

9,149,170,170,128,128,128,12
8,1825
A0 1124 DATA 128,128,128,255,25
5,128,128,128,149,149,170,17
0,1916
3B 1126 DATA 149,149,170,170,23
0,204,153,179,230,204,153,17
9,-1

```

## PROGRAM: CONDENSED SET

```

62 100 REM * CONDENSED *
8A 110 SA=62208
42 120 PRINT"ESC CODE FOR SET -
CHR$(27)CHR$("SA/256")"
23 130 B=0:FORX=0TO11:READA:B=B
+A:POKESA+X,A:NEXT:SA=SA+X:R
EADA:IFA-BTHEN130
04 140 IFA>0THENPRINT"DATA ERRO
R IN LINE"PEEK(63)+PEEK(64)*
256
40 1000 DATA 128,128,128,128,0,
0,0,0,128,128,223,128,1119
37 1002 DATA 0,0,0,0,135,128,13
5,128,0,0,0,0,526
2F 1004 DATA 255,148,255,128,0,
0,0,0,174,235,186,128,1509
D0 1006 DATA 0,0,0,0,243,136,23
1,128,0,0,0,0,738
DC 1008 DATA 255,217,166,208,0,
0,0,0,128,139,135,128,1376
41 1010 DATA 0,0,0,0,156,162,19
3,128,0,0,0,0,639
7E 1012 DATA 193,162,156,128,0,
0,0,0,170,156,170,128,1263
77 1014 DATA 0,0,0,0,136,190,13
6,128,0,0,0,0,590
82 1016 DATA 192,240,176,128,0,
0,0,0,136,136,136,128,1272
54 1018 DATA 0,0,0,0,128,176,17
6,128,0,0,0,0,608
99 1020 DATA 176,140,131,128,0,
0,0,0,191,161,191,128,1246
25 1022 DATA 0,0,0,0,164,191,16
0,128,0,0,0,0,643
26 1024 DATA 177,169,166,128,0,
0,0,0,161,165,155,128,1249
AD 1026 DATA 0,0,0,0,143,136,19
0,128,0,0,0,0,597
F0 1028 DATA 167,165,153,128,0,
0,0,0,156,166,153,128,1216
79 1030 DATA 0,0,0,0,129,177,14
1,131,0,0,0,0,578
BB 1032 DATA 191,165,191,128,0,
0,0,0,166,153,142,128,1264
E3 1034 DATA 0,0,0,0,128,148,14
8,128,0,0,0,0,552
D7 1036 DATA 128,212,180,128,0,
0,0,0,136,148,162,128,1222
63 1038 DATA 0,0,0,0,148,148,14
8,128,0,0,0,0,572
73 1040 DATA 162,148,136,128,0,
0,0,0,195,217,135,128,1249
3D 1042 DATA 0,0,0,0,190,217,21
7,222,0,0,0,0,846
1B 1044 DATA 186,170,190,128,0,
0,0,0,191,162,190,128,1345
4A 1046 DATA 0,0,0,0,190,162,16
2,128,0,0,0,0,642
D9 1048 DATA 190,162,191,128,0,
0,0,0,190,170,174,128,1333
41 1050 DATA 0,0,0,0,200,190,13
7,128,0,0,0,0,655

```

```

D1 1052 DATA 222,210,254,128,0,
0,0,0,191,130,190,128,1453
FB 1054 DATA 0,0,0,0,165,189,16
0,128,0,0,0,0,642
F3 1056 DATA 160,197,253,128,0,
0,0,0,191,136,182,128,1375
78 1058 DATA 0,0,0,0,161,191,16
0,128,0,0,0,0,640
9C 1060 DATA 190,140,190,128,0,
0,0,0,190,130,188,128,1284
D6 1062 DATA 0,0,0,0,190,162,19
0,128,0,0,0,0,670
2E 1064 DATA 254,162,190,128,0,
0,0,0,190,162,254,128,1468
C9 1066 DATA 0,0,0,0,190,130,13
4,128,0,0,0,0,582
A0 1068 DATA 174,170,186,128,0,
0,0,0,130,191,162,128,1269
12 1070 DATA 0,0,0,0,158,160,19
0,128,0,0,0,0,636
C9 1072 DATA 158,160,158,128,0,
0,0,0,190,152,190,128,1264
20 1074 DATA 0,0,0,0,182,136,18
2,128,0,0,0,0,628
2F 1076 DATA 222,208,254,128,0,
0,0,0,178,170,166,128,1454
21 1078 DATA 0,0,0,0,255,193,12
8,128,0,0,0,0,704
70 1080 DATA 164,190,165,128,0,
0,0,0,128,193,255,128,1351
3B 1082 DATA 0,0,0,0,132,191,13
2,128,0,0,0,0,583
16 1084 DATA 136,156,170,136,0,
0,0,0,128,128,128,128,1110
D8 1086 DATA 0,0,0,0,191,133,19
1,128,0,0,0,0,643
42 1088 DATA 191,165,187,128,0,
0,0,0,191,161,161,128,1312
18 1090 DATA 0,0,0,0,191,161,15
8,128,0,0,0,0,638
39 1092 DATA 191,165,165,128,0,
0,0,0,191,133,133,128,1234
4C 1094 DATA 0,0,0,0,191,161,18
5,128,0,0,0,0,665
7A 1096 DATA 191,132,191,128,0,
0,0,0,161,191,161,128,1283
E6 1098 DATA 0,0,0,0,176,161,19
1,128,0,0,0,0,656
52 1100 DATA 191,132,187,128,0,
0,0,0,191,160,160,128,1277
92 1102 DATA 0,0,0,0,191,134,19
1,128,0,0,0,0,644
26 1104 DATA 191,129,190,128,0,
0,0,0,158,161,158,128,1243
31 1106 DATA 0,0,0,0,191,137,14
3,128,0,0,0,0,599
62 1108 DATA 158,161,254,192,0,
0,0,0,191,137,183,128,1404
A7 1110 DATA 0,0,0,0,167,165,18
9,128,0,0,0,0,649
00 1112 DATA 129,191,129,128,0,
0,0,0,159,160,191,128,1215
75 1114 DATA 0,0,0,0,159,160,15
9,128,0,0,0,0,606
6B 1116 DATA 191,152,191,128,0,
0,0,0,187,132,187,128,1296
3C 1118 DATA 0,0,0,0,135,188,13
5,128,0,0,0,0,586
61 1120 DATA 185,165,163,128,0,
0,0,0,0,0,0,0,0,641
E5 1122 DATA 0,0,0,0,0,0,0,0,0,
0,0,0,0
8B 1124 DATA 0,0,0,0,0,0,0,0,0,
0,0,0,0
C5 1126 DATA 0,0,0,0,0,0,0,0,0,
0,0,0,-1

```



00	Ø	@	p	-	P
	16	32	48	64	80
01	!	a	q	A	Q
	17	33	49	65	81
02	"	b	r	B	R
	18	34	50	66	82
03	#	c	s	C	S
	19	35	51	67	83
04	\$	d	t	D	T
	20	36	52	68	84
05	%	e	u	E	U
	21	37	53	69	85
06	&	f	v	F	V
	22	38	54	70	86
07	'	g	w	G	W
	23	39	55	71	87

(	8	h	x	H	X
08	24	40	56	72	88
)	9	i	y	I	Y
09	25	41	57	73	89
*	:	j	z	J	Z
10	26	42	58	74	90
+	;	k	[	K	+
11	27	43	59	74	91
,	<	l	£	L	☒
12	28	44	60	75	92
-	=	m	]	M	
13	29	45	61	76	93
.	>	n	↑	N	☐
14	30	46	62	78	94
/	?	o	←	O	☐
15	31	47	63	79	95

The above shows the order in which the information defining the characters is stored. This order has been chosen because it makes the easiest (therefore quickest) conversion for the driving software.

**NOTE:** The characters from 96 to 127 cannot be re-defined. The driving software prints the normal MPS 803 characters. The characters no.96 to 127 are those printed with CHR\$(224) to CHR\$(255).

## C64 PLUS DISK

## C64

## NO LOAD DIRECTORY

1/1

When you have produced a program it is wise to prevent other people from reading your disks to see what is on them. The routine presented here will allow you to stop the disk directory from being loaded into the computers memory.

The routine forces the disk drive to keep on reading the directory, without actually loading it.

P.A.Eves

```

10 PRINT "INSERT DISK - ANY KEY
   TO CONTINUE"
20 POKE198,0:WAIT198,1
30 OPEN1,8,15,"IO:":OPENS,8,5,"#
   ":T=18:S=1:QS=CHR$(C)
40 PRINT#1,"U1 5 0 "T;S
50 FORI=0TO1:PRINT#1,"M-R"CHR$(I
   )CHR$(5):GET#1,B$:B4=B$+QS:B(I)=
   ASC(B$):NEXT
60 IFB(0)=18THENS=B(1):GOTO40
70 PRINT#1,"B-P:5 0":PRINT#5,CHR
   $(I);CHR$(1);
80 PRINT#1,"U2 5 0 "T;S:INPUT#1,
   A,A$:PRINTA$:CLOSE1:CLOSE5

```

# YC WRITER

*Sell your typewriter and get into serious wordprocessing with with 'YC Writer', an 80 column wordprocessor.*

By Burghard-Henry Lehmann

**P**eople who know nothing of the joys and tribulations of programming a computer and who are not interested in arcade games often ask, rather cynically, "What good are computers anyhow?"

Of course, they do not question the use of the kind of computer the gas board, for example, employs. The usefulness of this becomes clear every quarter when they get their gas bill. But what good is a home computer?

Well, I believe there is one good, solid reason why nearly everyone should invest in a home computer: wordprocessing. Everybody who has to do any writing at all, be it for work or for pleasure can benefit tremendously from using a wordprocessor. Even if all you want to do is write some letters, you will not know how easy and carefree it can be until you have done it on a wordprocessor.

A wordprocessor is more than just a souped-up typewriter or even an

electronic typewriter. It should really be called a 'text processor', because a good wordprocessing program goes far beyond letting you enter mere 'words'. It allows you to build up a piece of text and then restructure it in any way you like. And all this without wasting a single sheet of paper.

No more second, third and fourth drafts! You start by writing from the top of your head and correct the text as you go along. A wordprocessor allows you to develop a letter, an article or even a novel from its inception to its final form all in one go without wasting time on rewriting manuscript pages which have come to look like battlefields. The savings in time and material are tremendous!

## Getting Started

There is no better way to find out about wordprocessing than doing it. This is

what I have written *YC Writer* for, to give you a very real taste of it.

The first thing you will notice when the program has started is that the letters are much smaller than the ordinary C 64 letters. This is because *YC Writer* uses a sort of microprint which is printed on the high-resolution screen and gives you 80 characters per screen row.

This is of course the number of characters you get on any of the Commodore printers. So the main advantage of *YC Writer* is that you'll get on paper exactly what you see on screen.

This kind of microprint may take a while to get used to, depending on the kind of TV set you've got. If you are unhappy with the way things are and find it an excessive strain on your eyes, do a bit of experimenting with different colours and also with different brightness and contrast settings on your TV.

To experiment with different



background and foreground colours hold down the CTRL-key and press "M". Now you will be prompted to enter the border, paper and ink colours you prefer. Type the number of the colour you want (e.g. 6 for blue — see your Commodore manual) and the colour will be changed immediately. Finally, if you are satisfied with your settings, press "Y" to return to the text, if not, press "N" and the process will be repeated.

## Entering Commands

Most commands to the program are given with the CTRL key held down and a single letter being entered, e.g. CTRL+L for "LOAD textfile", CTRL+S for "SAVE textfile" and so on.

## Help

If you press function key 1 you will be presented with the first of the two help pages which the program incorporates. The RETURN key gets you the second help page and lets you toggle between the two pages. Function key 1 returns you to the text.

Remember, all of the letters given with functions are to be entered with CTRL held down!

## Information On Screen

The first three lines at the top are reserved for information. First you get the number of the line and the number of the column the cursor is on at any moment. Enter a few words and you'll see what I mean.

Next to it you get the number of words you have written so far. This wordcount is updated as you write. Later on when you start editing text it soon gets out of date. So, to get the exact wordcount press CTRL+U. This will update the number of words contained in the whole of the textfile.

Next to the number of words in the top line you see a "W", a "J" and a space in between four stars. These letters tell you which text entry mode is switched on. "W" stands for "word wrap" and "J" stands for "right hand justification". More about this and the space next to it in a minute.

The line below gives you the name of the document you are writing. When you enter the program this has the default

name "no name". You can change this to a 16-letter name of your own choice by pressing CTRL+N. Now the cursor will move into the right position, ready for you to enter the document name.

There is a practical reason for this: This name will also be the filename used later on when you want to SAVE your document onto disk or tape.

## The Tabulator

The third line at the top of the screen also has a practical reason beyond mere cosmetics: It shows you the tab position on each line.

To start with there is a tab point every 5 characters. Press function key 5 and the cursor will jump forward to the next tab position. Press function key 6 and the cursor will jump backwards to the former tab position.

You can install your own tab-position anywhere on the line by pressing CTRL+T. A "/" appears on the tab-line at the top of the screen where the new tab is. Press CTRL+T again, and the "/" vanishes.

If you want a completely different set-up of tab-positions than the one given press CTRL+F. Now all the tab-points are erased and, with CTRL+T you can make up your own tab-spacings. Press CTRL+F again and the default tab-positions are restored.

## Entry Modes

There are really two sides to wordprocessing: you want to enter text and you want to edit it in the most convenient manner possible.

For this YC Writer has three entry modes: word wrap, right hand justification and insert.

Word wrap means that you can write your text as if you had one long continuous line. That is, you can ignore the end of a line and the computer does the rest. If you start a word at the end of the line it will automatically be moved onto the new line while you are writing.

For this to work there is an extra keystroke at the beginning of each line: If you enter a letter at the beginning of a new line the computer will know that this is part of a word started on the previous line and will move the whole word onto the new line. If you enter a

space, the cursor won't move on, because the computer knows that the characters on the end of the line above form a complete word and are not to be moved (or "word wrapped") onto the new line. All this works of course only if you have the word wrap mode switched on. When the program starts, you will find it is on, but you can switch it on or off by pressing CTRL+W.

## Right Justification

The next important entry feature is right hand justification. This always works in combination with word wrap and means that the line is spaced out in such a way that it is flush with the right hand side. Like word wrap it works automatically as you write.

Again, you can turn right hand justification on or off by pressing CTRL+J. But note: You can have word wrap without justification to get the 'typewriter look', but you can't have justification without word wrap.

## Insert Mode

The third entry mode is most useful when you want to edit the text you have written and add additional words or whole sentences.

For this move the cursor onto the paragraph into which you want to insert something. Then press CTRL+I.

Now the paragraph is reformatted by the computer, that is, it is 'un-justified' so that there is at least one space at the end of each line. This is necessary for insert to work properly. Don't worry about this restructuring of the paragraph! After you have done your insert and switched insert off again (as you should every time!) the whole paragraph will automatically be word wrapped and justified again!

Once in the insert mode you can enter text, but it will not overwrite other text. Instead the text to the right will be pushed along by the cursor. If there isn't enough space at the end of the paragraph it will automatically insert an empty line. So you can insert as much as you like.

Do remember to switch insert off after you have finished! Otherwise it will go on wherever you put the cursor.

If you want an empty line anywhere, you can insert one by pressing CTRL+A.

This will move the rest of the textfile down by one line.

## Erasing

Conversely, you can erase the line the cursor is on by pressing CTRL+B. This moves the rest of the textfile into the line you want to be erased.

If you want simply to erase one or two characters use the delete key as normal.

If you want to erase a whole block of text quickly and efficiently there is a powerful block erase facility. For this you first have to tell the computer the first line of the block you want to be erased and then the last line.

This is called marking out a block, and I mention it especially because the same procedure will also be used for marking out a block which you want to be moved or copied. It works like this:

## Block-set

Move the cursor onto the first line of the block you want to mark out. Press CTRL+G. You will notice that in the information header at the top of the screen a remark has appeared, for example: "Blk-start: 4". This is to remind you that you have marked out the beginning of a block starting at line 4.

Next move the cursor to the last line of the block you want to mark out and press again CTRL+G. Now the message "Blk-end: 10" will appear at the top of the screen.

You have now set a block starting at line 4 and ending at line 10, inclusive. This is now the "current block". Thus a block always goes from the beginning of one line to the end of another.

If for any reason you are not happy with the parameters you have given press CTRL+G again and the info at the top of the screen will be erased so that you can start again.

To erase the block you have marked out, simply press CTRL+K.

If you want to get rid of the whole textfile and start afresh press CTRL+E. Since this is a pretty final command there is a safety-catch built into it: You will be asked if you are sure about erasing everything. If you are not, press "N" and no harm will be done. If you are certain, press "Y" and not only will the whole

of the textfile be erased but the program will reset as if you just have started it off.

## Moving and Copying

If you want to move the current block to somewhere else in the textfile, move the cursor to the line above the one you want the block moved to and press CTRL+O.

Similarly, if you want to copy the block: bring the cursor to the line you want and press CTRL+H.

You will notice that after block erasure and block move the messages at the top of the screen will vanish. Not so after block-copy! This is so that you can copy the block you have chosen as many times as you wish. But this only works of course, as long as you don't do any more editing. If you do, the position of the block may have changed so that you will get something different copied out!

## Formating Text

At any given time you can reformat a paragraph to have it right hand justified or not. CTRL+C un-justifies the paragraph the cursor is on, while CTRL+D justifies it.

For all this, and the insert mode, to work properly the computer has to know where a paragraph starts and ends. So there is an important rule: A paragraph has to be started with an indent of at least two spaces!

Other wordprocessors use formating characters to mark out the beginning or the end of a paragraph. With *YC Writer* I wanted to have no distracting formating characters on screen. For this to work, you have to obey the above rule. A small price to pay, don't you agree?

## Margins

Impressive as 80-columns on screen and on paper are, with most Commodore printers it looks rather cramped because it fills nearly the whole width of an A4 sheet. This doesn't look very good if you are writing a letter you want to create a good impression.

For this reason I was determined to include a margin setting facility in *YC Writer*. It only works on fresh textfile before you have entered any text. You have to stick with the margins you have chosen throughout the textfile and can't change

them afterwards.

Let's say you want a left hand margin of 10 characters. Put the cursor into the right position (2 tab-positions with F5) and press CTRL+X.

The margin is demonstrated graphically by the space on the left being filled and by the cursor position becoming column 0.

If you now want to set a margin of 10 characters to the right, again place the cursor at the appropriate position and press CTRL+Y. A similar thing will happen.

For technical reasons there is a rule (more rules!) to all this: The left margin has to be set on an even numbered column, while the right margin has to be set on an odd numbered column!

## Saving and Loading

Once you have written your document the first thing you want to do it is SAVE it on disk or tape.

It is a good idea to do this at regular intervals. Powercuts, however brief, are not an unknown thing and it takes only a few cycles of no electricity and hours of your work may be down the drain!

Saving a document is very straight forward. After you have given it the name you want as I have already described: Press CTRL+S.

In order to make the program work for tape as well as disk, so you don't always have to answer the annoying question: "Tape or Disk?", you can switch the program into the tape or disk mode by POKEing location 8010 from Basic and then saving that version.

As a matter of fact, you only have to do this POKE if you are using tape. Simply POKE 8010,1. If, for any reason, you want to revert to disk, POKE 8010,8.

The disk version of the program includes a replace facility. You can use this if you have already saved a certain document and want merely to replace it with a changed version.

Since there is a lot of discussion going on in the C 64 community about the safety of the replace facility and I seem to belong to the 2 percent of disk-drive owners where it isn't safe to use, I have circumnavigated this sordid area by doing the replace in *YC Writer* with a combination of scratch and SAVE. Better safe than sorry!



## Printout

Finally, you will naturally want your document printed out.

This is very easy with *YC Writer*: Simply get your printer switched on and ready and then press CTRL+P. The whole of your document will be sent to the printer as it appears on screen.

Many wordprocessors have additional print options, like page-numbers, footers, headers and so on. I didn't have enough time in developing this program to give you anything but a straight forward printout. I also thought that this is a nice little programming job you could easily do yourself in Basic.

Incidentally, you can quit the program at any given moment by pressing function key 7. And you can re-enter it by entering SYS 8050.

There is some 8k of memory empty for your own programs. The textfile starts in memory at location 16020.

There textfile is limited to 300 lines. Not quite enough for 'War and Peace', but remember Tolstoy's textfile was limited to one sheet of foolscap and just think of all his corrections!

## Getting it all in

Entering the programs.

- 1) Type in each of the programs presented here separately using the SYNTAX CHECKER found on the LISTINGS page.
- 2) If using cassette SAVE WRITER BOOT on a separate cassette to the other programs.
- 3) LOAD and RUN YC WRITER A & B. When B has finished it will SAVE a new program out. If using tape this should be SAVED after WRITER BOOT.
- 4) LOAD and RUN YC WRITER C to YC WRITER D. When finished a second program will be SAVED. If using tape this should be SAVED after the program from 3.
- 5) To RUN the program simply LOAD and RUN WRITER BOOT. This will LOAD the other two parts and start the program.

### PROGRAM: YC WRITER A

```

2F 10 BL=34 :LN=50 :SA=4915
2
5B 20 FOR L=0 TO BL:GX=0:FOR D=
0 TO 15:READ A:GX=GX+A:POKE
SA+L*16+D,A:NEXT D
AS 30 READ A:IF A>GX THENPRINT
"ERROR IN LINE":LN+(L*10):ST
OP
40 40 NEXT L:END
77 50 DATA 173,237,3,201,2,240,
3,76,120,31,56,173,233,3,237
,235,2023
4C 60 DATA 3,133,181,173,234,3,
237,236,3,133,182,230,181,20
8,2,230,2369
1B 70 DATA 182,165,167,133,88,1
33,169,133,179,56,165,166,22
9,172,133,87,2357
AC 80 DATA 133,168,133,178,176,
6,198,88,198,169,198,179,24,
165,178,109,2300
50 90 DATA 58,31,133,178,144,2,
230,179,56,173,64,31,229,178
,133,73,1892
35 100 DATA 173,65,31,229,179,5
,73,144,14,24,165,178,109,57
,31,133,1610
43 110 DATA 178,144,229,230,179
,176,225,56,165,178,229,87,1
33,92,165,179,2645
78 120 DATA 229,88,133,93,24,16
5,168,101,181,133,89,165,169
,101,182,133,2154
31 130 DATA 90,32,0,35,32,37,19
3,144,18,24,173,235,3,101,18
1,133,1431
57 140 DATA 87,173,236,3,101,18
2,133,88,76,165,192,173,235,
3,133,87,2067
AD 150 DATA 173,236,3,133,88,16
5,168,133,89,165,169,133,90,
165,181,133,2224
F5 160 DATA 92,165,182,133,93,3
2,0,35,56,173,64,31,229,168,
133,73,1659
26 170 DATA 173,65,31,229,169,5
,73,176,18,24,165,168,101,18
1,141,64,1783
70 180 DATA 31,165,169,101,182,
141,65,31,76,236,192,24,173,
64,31,101,1782
68 190 DATA 181,141,64,31,173,6
5,31,101,182,141,65,31,32,37
,193,144,1612
E1 200 DATA 34,24,173,235,3,101
,181,141,235,3,173,236,3,101
,182,141,1966
D6 210 DATA 236,3,24,173,233,3,
101,181,141,233,3,173,234,3,
101,182,2024
50 220 DATA 141,234,3,173,232,3
,240,3,76,79,37,165,171,133,
91,32,1813
15 230 DATA 180,45,76,120,31,56
,173,235,3,229,168,133,73,17
3,236,3,1934
40 240 DATA 229,169,5,73,96,32,
145,178,200,192,16,208,249,3
2,90,193,2107
52 250 DATA 169,13,32,210,255,3

```

```

2,210,255,96,69,78,84,69,82,
32,70,1756
ED 260 DATA 73,76,69,78,65,77,6
9,58,32,0,169,0,76,98,193,76
,1209
86 270 DATA 164,193,32,68,229,2
4,162,12,160,0,32,240,255,16
9,147,160,2047
53 280 DATA 193,32,30,171,169,9
7,133,178,169,31,133,179,160
,0,169,32,1876
80 290 DATA 145,178,200,192,16,
208,249,32,164,193,169,13,32
,210,255,32,2288
53 300 DATA 210,255,96,69,78,84
,69,82,32,70,73,76,69,78,65,
77,1483
A7 310 DATA 69,58,32,0,169,0,13
3,204,70,207,32,228,255,240,
245,201,2143
64 320 DATA 17,240,241,201,145,
240,237,72,56,165,211,233,16
,168,104,166,2512
EE 330 DATA 211,224,16,240,50,1
44,48,192,16,176,25,201,13,2
40,61,32,1889
C0 340 DATA 210,255,201,20,240,
47,201,32,144,202,201,129,14
4,2,73,128,2229
C5 350 DATA 145,178,208,192,32,
228,255,240,251,201,13,240,3
1,201,20,240,2675
DE 360 DATA 222,201,157,240,218
,208,237,201,13,240,17,201,2
0,240,165,201,2781
8D 370 DATA 157,240,161,208,202
,136,169,32,145,178,208,152,
162,1,134,204,2489
3E 380 DATA 202,134,207,96,0,0,
0,0,0,0,0,0,0,0,0,0,639
D4 390 DATA 0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0

```

### PROGRAM: YC WRITER B

```

C0 10 BL=152 :LN=50 :SA=4976
9
5B 20 FOR L=0 TO BL:GX=0:FOR D=
0 TO 15:READ A:GX=GX+A:POKE
SA+L*16+D,A:NEXT D
AS 30 READ A:IF A>GX THENPRINT
"ERROR IN LINE":LN+(L*10):ST
OP
AC 40 NEXT L:PRINT"[DOWN]GET R
EADY TO SAVE AND PRESS A KEY
":POKE198,0:WAIT198,1
A2 45 POKE43,0:POKE44,192:POKE4
5,27:POKE46,204:X=PEEK(186):
SAVE"WRITER1",X
69 50 DATA 120,32,62,196,32,230
,195,32,10,196,169,0,141,192
,2,169,1778
51 60 DATA 165,141,20,3,169,194
,141,21,3,169,235,141,38,3,1
69,197,1809
61 70 DATA 141,39,3,32,152,195,
88,96,173,192,2,240,3,76,41,
195,1668
86 80 DATA 206,184,2,240,79,169
,50,205,184,2,240,3,76,41,19
5,169,2045
1F 90 DATA 1,141,186,2,165,251,
141,188,2,165,252,141,189,2,
173,167,2166

```

BA	100 DATA 2,141,190,2,133,251,173,168,2,141,191,2,133,252,32,214,2027	06	360 DATA 173,2,24,14,172,2,4,6,173,2,24,173,172,2,101,251,133,1464	46	620 DATA 141,186,2,173,190,2,133,251,173,191,2,133,252,160,7,185,2181
96	110 DATA 195,160,7,177,251,1,53,193,2,174,171,2,208,5,9,2,40,76,2023	77	370 DATA 251,173,173,2,101,2,52,133,252,173,167,2,133,253,173,168,2,2408	2C	630 DATA 193,2,145,251,136,1,6,248,169,51,141,184,2,173,1,72,2,201,2086
68	120 DATA 241,194,9,15,145,25,1,136,16,234,32,222,195,169,20,141,184,2204	AE	380 DATA 133,254,173,171,2,2,08,68,160,0,162,4,177,253,41,15,141,1962	69	640 DATA 255,208,8,169,126,1,41,172,2,76,79,198,201,224,1,44,9,56,2068
CC	130 DATA 2,76,31,195,169,0,1,41,186,2,173,190,2,133,251,1,73,191,1915	21	390 DATA 172,2,177,251,41,24,0,13,172,2,145,253,230,253,2,08,2,230,2391	11	650 DATA 233,64,141,172,2,76,79,198,201,192,144,6,56,233,96,141,2034
A6	140 DATA 2,133,252,160,7,185,193,2,145,251,136,16,248,16,9,70,141,2110	FC	400 DATA 254,177,253,41,15,1,41,172,2,177,251,10,10,10,10,13,172,1708	28	660 DATA 172,2,201,13,208,6,32,246,201,76,238,201,201,17,208,117,2139
25	150 DATA 184,2,173,188,2,133,251,173,189,2,133,252,32,23,4,255,165,2368	16	410 DATA 2,145,253,230,253,2,08,2,230,254,230,251,208,2,2,30,252,202,2952	DB	670 DATA 173,168,2,201,254,1,44,73,173,167,2,201,0,144,66,173,171,2112
E9	160 DATA 204,208,41,198,205,208,37,169,20,133,205,164,21,1,70,207,174,2454	6A	420 DATA 208,201,238,171,2,3,2,222,195,76,35,197,160,0,16,2,4,177,2080	03	680 DATA 2,141,178,2,173,167,2,141,174,2,173,168,2,141,1,75,2,1643
01	170 DATA 135,2,177,209,176,1,7,230,207,133,206,32,36,234,177,243,141,2355	DS	430 DATA 253,41,240,141,172,2,177,251,74,74,74,74,13,172,2,145,1905	68	690 DATA 173,169,2,141,176,2,173,170,2,141,177,2,32,80,1,97,173,1810
BF	180 DATA 135,2,174,134,2,165,206,73,128,32,28,234,165,1,41,16,1536	22	440 DATA 253,230,253,208,2,2,30,254,177,253,41,240,141,17,2,2,177,251,2884	4A	700 DATA 178,2,141,171,2,173,174,2,141,167,2,173,175,2,1,41,168,1812
9C	190 DATA 240,10,160,0,132,19,2,165,1,9,32,208,8,165,192,2,08,6,1728	B3	450 DATA 41,15,13,172,2,145,253,230,253,208,2,230,254,23,0,251,208,2507	77	710 DATA 2,173,176,2,141,169,2,173,177,2,141,170,2,76,20,7,198,1811
89	200 DATA 165,1,41,31,133,1,3,2,135,234,202,189,119,2,201,140,208,1834	FE	460 DATA 2,230,252,202,208,2,01,165,253,141,167,2,165,254,141,168,2,2553	A6	720 DATA 24,173,167,2,105,64,141,167,2,173,168,2,105,1,1,41,168,1603
24	210 DATA 23,198,198,173,17,2,08,41,32,208,6,32,152,195,76,149,195,1903	16	470 DATA 206,171,2,32,222,19,5,173,169,2,133,251,173,170,2,133,252,2286	69	730 DATA 2,24,173,169,2,105,40,141,169,2,173,170,2,105,0,141,1418
CC	220 DATA 32,182,195,169,14,3,2,202,241,76,126,234,173,2,2,21,9,3,1911	AE	480 DATA 160,0,173,134,2,10,10,10,10,141,172,2,177,251,4,1,15,1308	8C	740 DATA 170,2,76,238,201,20,1,19,208,6,32,62,196,76,238,201,201,2127
C5	230 DATA 141,2,221,173,17,20,8,9,32,141,17,208,173,0,221,41,252,1856	CE	490 DATA 13,172,2,145,251,17,3,171,2,208,8,238,169,2,208,3,238,2003	05	750 DATA 20,240,3,76,248,199,173,167,2,141,176,2,173,168,2,141,1931
6A	240 DATA 141,0,221,169,56,14,1,24,208,96,173,2,221,9,3,14,1,2,1607	6F	500 DATA 170,2,96,169,0,133,251,24,105,64,133,253,169,22,4,133,252,2178	93	760 DATA 177,2,173,169,2,141,178,2,173,170,2,141,179,2,3,2,246,1789
33	250 DATA 221,173,17,208,41,2,23,141,17,208,173,0,221,41,2,52,9,3,1948	C6	510 DATA 105,1,133,254,32,21,4,195,160,0,177,253,145,251,165,253,201,2539	72	770 DATA 201,173,176,2,141,1,67,2,173,177,2,141,168,2,173,178,2,1878
1E	260 DATA 141,0,221,169,21,14,1,24,208,96,120,165,1,41,253,133,1,1735	45	520 DATA 63,208,6,165,254,20,1,255,240,15,230,251,208,2,2,30,252,230,2810	AE	780 DATA 141,169,2,173,179,2,141,170,2,162,8,160,0,32,21,4,195,1750
94	270 DATA 96,165,1,9,2,133,1,88,96,169,0,133,251,169,224,133,1670	36	530 DATA 253,208,2,230,254,7,6,102,197,32,222,195,169,0,2,30,251,208,2629	F3	790 DATA 173,171,2,208,77,17,3,167,2,201,0,208,13,173,168,2,201,1939
52	280 DATA 252,160,0,169,0,145,251,165,251,201,63,208,7,16,5,252,201,2490	94	540 DATA 2,230,252,145,251,1,66,251,224,63,208,6,166,252,224,255,240,2935	4D	800 DATA 224,208,6,32,222,19,5,76,238,201,56,173,169,2,23,3,1,141,2177
39	290 DATA 255,208,1,96,230,25,1,208,2,230,252,76,240,195,1,69,0,133,2546	EB	550 DATA 3,76,138,197,169,0,133,251,24,105,40,133,253,16,9,204,133,2028	66	810 DATA 169,2,173,170,2,233,0,141,170,2,56,173,176,2,23,3,8,1710
99	300 DATA 251,169,204,133,252,160,0,173,33,208,41,15,133,253,173,134,2332	C5	560 DATA 252,105,0,133,254,1,77,253,145,251,165,253,201,2,32,208,6,165,2800	9A	820 DATA 141,176,2,141,167,2,173,177,2,233,0,141,177,2,1,41,168,1843
6B	310 DATA 2,10,10,10,10,41,24,0,5,253,145,251,166,251,224,232,208,2058	22	570 DATA 254,201,207,240,15,230,251,208,2,230,252,230,25,3,208,2,230,3013	E4	830 DATA 2,173,176,2,133,253,173,177,2,133,254,177,253,4,1,240,141,2330
F1	320 DATA 7,166,252,224,207,2,08,1,96,230,251,208,2,230,25,2,76,38,2448	3E	580 DATA 254,76,178,197,169,0,141,167,2,169,254,141,168,2,169,192,2279	FB	840 DATA 182,2,173,174,2,133,251,173,175,2,133,252,140,1,72,2,56,2022
F2	330 DATA 196,169,0,141,167,2,169,224,141,168,2,169,0,141,169,2,1860	CF	590 DATA 141,169,2,169,207,1,41,170,2,169,0,141,171,2,96,141,172,1893	C9	850 DATA 165,251,233,8,133,2,51,165,252,233,0,133,252,177,251,24,46,2574
D4	340 DATA 169,204,141,170,2,1,69,0,141,171,2,96,169,0,141,173,2,1750	CF	600 DATA 2,72,138,72,152,72,165,154,201,3,240,8,104,168,104,170,1825	A1	860 DATA 172,2,42,46,172,2,4,2,46,172,2,42,46,172,2,42,46,1048
08	350 DATA 32,214,195,169,123,133,251,169,202,133,252,24,1,4,172,2,46,2131	4E	610 DATA 104,76,202,241,169,1,141,192,2,173,186,2,240,30,169,0,1928	CB	870 DATA 172,2,145,251,46,17,2,2,46,172,2,46,172,2,46,172,2,1450



```

6C 880 DATA 165,251,205,176,2,2
    08,200,165,252,205,177,2,208
    ,193,173,171,2753
D7 890 DATA 2,208,9,177,253,41,
    15,13,182,2,145,253,238,174,
    2,208,1922
9A 900 DATA 3,238,175,2,238,176
    ,2,208,3,238,177,2,202,240,1
    1,173,2088
8F 910 DATA 171,2,240,3,76,111,
    199,76,94,199,32,222,195,206
    ,171,2,1999
7C 920 DATA 240,6,238,171,2,238
    ,171,2,76,238,201,201,29,208
    ,59,173,2253
B7 930 DATA 171,2,208,6,238,171
    ,2,76,238,201,206,171,2,24,1
    73,167,2056
A2 940 DATA 2,105,8,141,167,2,1
    73,168,2,105,0,141,168,2,238
    ,169,1591
F9 950 DATA 2,208,3,238,170,2,1
    73,167,2,201,64,208,10,173,1
    68,2,1791
4D 960 DATA 201,255,208,3,32,80
    ,197,76,238,201,201,141,208,
    6,32,246,2325
AB 970 DATA 201,76,238,201,201,
    145,208,53,162,225,236,168,2
    ,144,9,208,2477
5E 980 DATA 41,174,167,2,224,64
    ,144,34,56,173,167,2,233,64,
    141,167,1853
4C 990 DATA 2,173,168,2,233,1,1
    41,168,2,56,173,169,2,233,40
    ,141,1704
AB 1000 DATA 169,2,173,170,2,23
    3,0,141,170,2,76,238,201,201
    ,147,208,2133
36 1010 DATA 12,32,230,195,32,1
    0,196,32,62,196,76,238,201,2
    01,148,240,2101
88 1020 DATA 3,76,114,201,173,1
    67,2,141,176,2,173,168,2,141
    ,177,2,1718
5B 1030 DATA 173,169,2,141,178,
    2,173,170,2,141,179,2,32,246
    ,201,173,1984
EF 1040 DATA 176,2,141,167,2,17
    3,177,2,141,168,2,173,178,2,
    141,169,1814
23 1050 DATA 2,173,179,2,141,17
    0,2,173,167,2,133,251,173,16
    8,2,133,1871
3A 1060 DATA 252,160,0,140,172,
    2,162,8,32,214,195,173,171,2
    ,240,9,1932
7D 1070 DATA 177,251,141,182,2,
    41,15,145,251,177,251,24,110
    ,172,2,106,2047
08 1080 DATA 110,172,2,106,110,
    172,2,106,110,172,2,106,110,
    172,2,145,1599
5F 1090 DATA 251,110,172,2,110,
    172,2,110,172,2,110,172,2,24
    ,165,251,1827
7D 1100 DATA 105,8,133,251,165,
    252,105,0,133,252,165,251,20
    5,174,2,208,2409
F4 1110 DATA 200,165,252,205,17
    5,2,208,193,173,171,2,240,17
    ,173,176,2,2354
24 1120 DATA 133,251,173,177,2,
    133,252,173,182,2,41,240,145
    ,251,238,176,2569
C7 1130 DATA 2,208,3,238,177,2,
    173,176,2,133,251,173,177,2,

```

```

133,252,2102
2C 1140 DATA 238,174,2,208,3,23
8,175,2,140,172,2,173,171,2,
240,9,1949
09 1150 DATA 177,251,141,182,2,
41,15,145,251,202,240,3,76,2
30,200,32,2188
DD 1160 DATA 222,195,76,238,201
,201,157,208,68,173,171,2,24
0,6,206,171,2535
4E 1170 DATA 2,76,238,201,173,1
67,2,201,0,208,10,173,168,2,
201,224,2046
08 1180 DATA 208,3,76,238,201,2
38,171,2,56,173,167,2,233,8,
141,167,2084
BA 1190 DATA 2,173,168,2,233,0,
141,168,2,56,173,169,2,233,1
,141,1664
36 1200 DATA 169,2,173,170,2,23
3,0,141,170,2,76,238,201,201
,160,208,2146
C6 1210 DATA 5,169,32,141,172,2
,201,161,144,5,169,128,141,1
72,2,201,1845
4A 1220 DATA 32,144,30,201,129,
176,26,56,233,32,141,172,2,3
2,88,196,1690
2A 1230 DATA 173,167,2,201,64,2
08,10,173,168,2,201,255,208,
3,32,80,1947
42 1240 DATA 197,169,0,141,192,
2,76,249,197,169,0,141,174,2
,169,224,2102
E2 1250 DATA 141,175,2,162,0,17
3,175,2,205,168,2,144,12,208
,31,173,1773
26 1260 DATA 174,2,205,167,2,24
0,2,176,21,24,173,174,2,105,
64,141,1672
54 1270 DATA 174,2,173,175,2,10
5,1,141,175,2,232,76,2,202,1
73,172,1807
A6 1280 DATA 2,201,148,240,72,2
01,20,240,68,173,174,2,141,1
67,2,173,2024
77 1290 DATA 175,2,141,168,2,16
9,0,141,171,2,169,0,141,169,
2,169,1621
09 1300 DATA 204,141,170,2,232,
202,240,20,24,173,169,2,105,
40,141,169,2034
15 1310 DATA 2,173,170,2,105,0,
141,170,2,76,82,202,173,167,
2,201,1668
E6 1320 DATA 64,144,10,173,168,
2,201,255,208,3,32,80,197,96
,0,0,1633
26 1330 DATA 0,0,102,102,6,96,8
5,0,0,0,85,245,95,85,39,170,
1110
EB 1340 DATA 115,226,234,242,79
,87,4,164,169,112,36,0,0,0,1
8,68,1554
20 1350 DATA 68,33,66,17,17,36,
0,150,246,144,2,39,34,0,0,0,
852
28 1360 DATA 3,52,0,7,0,0,0,0,6
,96,17,34,68,136,105,189,713
78 1370 DATA 153,96,38,34,34,11
2,105,18,72,240,113,39,18,64
,19,121,1276
58 1380 DATA 241,16,116,71,17,9
6,18,78,153,96,241,34,68,64,
105,150,1564
DE 1390 DATA 153,96,105,151,18,

```

64, 6, 96, 6, 96, 6, 96, 6, 100, 18, 7  
2, 1089

49 1400 DATA 66, 16, 0, 112, 112, 0,  
132, 33, 36, 128, 37, 18, 32, 32, 10  
5, 187, 1046

F0 1410 DATA 136, 112, 0, 113, 117,  
112, 68, 117, 85, 112, 0, 116, 68, 1  
12, 17, 117, 1402

91 1420 DATA 85, 112, 0, 117, 116, 1  
12, 3, 68, 100, 64, 0, 117, 87, 23, 6  
8, 117, 1189

E8 1430 DATA 85, 80, 32, 34, 34, 32,  
32, 34, 34, 36, 69, 86, 101, 80, 68,  
68, 905

E1 1440 DATA 68, 96, 0, 159, 153, 14  
4, 0, 117, 85, 80, 0, 117, 85, 112, 0  
, 117, 1333

C7 1450 DATA 87, 68, 0, 117, 87, 17,  
0, 116, 68, 64, 0, 116, 113, 112, 68  
, 116, 1149

A9 1460 DATA 68, 112, 0, 85, 85, 112  
, 0, 85, 85, 32, 0, 153, 159, 144, 0,  
85, 1205

90 1470 DATA 37, 80, 0, 85, 87, 23, 0  
, 113, 36, 112, 118, 102, 102, 112,  
37, 79, 1123

48 1480 DATA 68, 240, 115, 51, 51, 1  
12, 2, 114, 34, 34, 0, 79, 64, 0, 0, 1  
5, 979

0C 1490 DATA 240, 0, 37, 87, 85, 80,  
101, 87, 85, 96, 117, 68, 69, 112, 1  
01, 85, 1450

05 1500 DATA 85, 96, 116, 71, 68, 11  
2, 116, 71, 68, 64, 117, 71, 85, 112  
, 85, 87, 1424

8A 1510 DATA 85, 80, 114, 34, 34, 11  
2, 114, 34, 42, 224, 85, 86, 101, 80  
, 68, 68, 1361

86 1520 DATA 68, 112, 159, 153, 153  
, 144, 157, 223, 187, 144, 117, 85,  
85, 112, 117, 87, 2103

0F 1530 DATA 68, 64, 117, 85, 85, 35  
, 117, 87, 101, 80, 116, 71, 17, 112  
, 114, 34, 1303

4B 1540 DATA 34, 32, 85, 85, 85, 112  
, 85, 85, 85, 32, 153, 153, 159, 144  
, 85, 82, 1496

5A 1550 DATA 37, 80, 85, 85, 34, 32,  
113, 18, 68, 112, 102, 111, 246, 10  
2, 165, 165, 1555

7D 1560 DATA 165, 165, 102, 102, 10  
2, 102, 0, 250, 170, 160, 255, 119,  
51, 17, 255, 255, 2270

47 1570 DATA 255, 255, 0, 0, 0, 0, 0,  
0, 0, 0, 0, 0, 0, 0, 0, 510

PROGRAM: YC WRITER C

```

14 10 BL=49 :LN=50 :SA=5731
5B 20 FOR L=0 TO BL: CX=0: FOR D=
0 TO 15: READ A: CX=CX+A: POKE
SA+L*16+D,A: NEXT D
AS 30 READ A: IF A<CX THEN PRINT
"ERROR IN LINE";LN+(L*10): ST
OP
40 40 NEXT L: END
3F 50 DATA 76,105,22,76,206,23,
32,206,23,160,0,140,232,3,13
2,181,1617
5E 60 DATA 132,182,200,140,238,
3,160,0,177,168,201,32,208,8

```

```

,200,204,2253
B0 70 DATA 58,31,208,244,240,93
,132,183,132,184,177,168,201
,32,208,38,2329
29 80 DATA 204,58,31,176,48,133
,73,200,177,168,201,32,208,2
2,132,183,2046
4F 90 DATA 164,184,145,168,164,
183,177,168,201,32,208,18,20
0,204,57,31,2304
7B 100 DATA 208,244,240,17,165,
73,164,184,145,168,230,183,2
30,184,164,183,2782
F2 110 DATA 204,57,31,208,197,1
73,232,3,208,103,164,184,204
,57,31,176,2232
B5 120 DATA 51,169,32,145,168,2
00,204,57,31,208,248,173,238
,3,240,3,2170
B0 130 DATA 32,186,45,24,165,16
8,109,57,31,133,168,144,2,23
0,169,230,1893
A7 140 DATA 91,160,0,177,168,20
1,32,208,7,200,177,168,201,3
2,240,3,2065
2B 150 DATA 76,121,22,96,173,68
,31,240,210,177,168,201,32,2
40,3,136,1994
3D 160 DATA 208,242,162,0,200,1
77,168,157,65,3,169,32,145,1
68,232,200,2328
FB 170 DATA 204,57,31,208,240,1
57,65,3,232,134,181,238,232,
3,76,222,2283
CC 180 DATA 22,164,184,204,58,3
1,176,10,169,32,145,168,200,
204,57,31,1855
46 190 DATA 208,248,165,184,133
,183,24,101,181,205,57,31,14
4,66,56,173,2159
60 200 DATA 58,31,229,181,168,1
77,168,201,32,240,3,136,208,
247,132,183,2394
CB 210 DATA 200,162,0,177,168,1
57,234,159,201,32,208,8,200,
177,168,201,2452
27 220 DATA 32,240,12,136,169,3
2,145,168,232,200,204,57,31,
208,228,169,2263
82 230 DATA 32,157,234,159,232,
134,182,24,165,183,101,181,1
33,184,208,2,2311
CF 240 DATA 133,184,164,183,177
,168,164,184,145,168,198,184
,198,183,16,242,2691
1C 250 DATA 166,181,164,184,202
,189,65,3,145,168,202,136,16
,247,166,182,2416
15 260 DATA 240,19,189,234,159,
157,65,3,202,16,247,165,182,
133,181,169,2361
95 270 DATA 0,133,182,240,3,206
,232,3,76,222,22,165,166,133
,168,165,2116
0E 280 DATA 167,133,169,56,165,
168,229,172,133,168,176,2,19
8,169,165,171,2441
B1 290 DATA 133,91,165,168,205,
69,31,208,7,165,169,205,70,3
1,240,32,1989
DB 300 DATA 160,0,177,168,201,3
2,208,7,200,177,168,201,32,2
40,17,56,2044
C0 310 DATA 165,168,237,57,31,1
33,168,176,2,198,169,198,91,
76,229,23,2121
FB 320 DATA 96,127,127,127,127,
127,127,127,127,127,127,127,
127,127,127,127,2001
6B 330 DATA 127,127,127,127,127
,127,127,127,127,127,127,127
,127,127,127,127,2032
3A 340 DATA 127,127,127,127,127
,32,102,22,169,0,133,181,133
,182,141,232,1962
09 350 DATA 3,165,168,133,73,16
5,169,133,74,24,165,73,109,5
7,31,133,1675
D1 360 DATA 73,144,2,230,74,160
,0,177,73,201,32,208,10,200,
177,73,1834
A4 370 DATA 201,32,208,3,238,23
2,3,172,57,31,177,168,201,32
,240,44,2039
6D 380 DATA 136,177,168,201,32,
240,37,173,68,31,240,32,162,
0,177,168,2042
04 390 DATA 201,32,240,4,232,13
6,208,246,134,181,162,0,200,
177,168,157,2478
6B 400 DATA 65,3,169,32,145,168
,200,232,228,181,208,241,173
,232,3,208,2488
89 410 DATA 3,32,20,30,32,186,4
5,24,165,168,109,57,31,133,1
68,144,1347
31 420 DATA 2,230,169,230,91,16
0,0,177,168,201,32,208,63,20
0,177,168,2276
C6 430 DATA 201,32,208,10,165,1
81,240,3,76,146,25,76,97,25,
160,0,1645
BF 440 DATA 177,168,201,32,208,
38,173,68,31,208,3,76,68,24,
132,184,1791
D7 450 DATA 200,132,183,174,57,
31,164,183,177,168,164,184,1
45,168,230,183,2543
09 460 DATA 230,184,202,208,241
,172,58,31,169,32,145,168,17
3,68,31,240,2352
5D 470 DATA 90,165,181,240,86,1
72,58,31,166,181,177,168,201
,32,208,79,2235
67 480 DATA 136,202,208,246,177
,168,201,32,208,3,136,208,24
7,132,183,24,2511
19 490 DATA 165,183,101,181,133
,184,164,183,177,168,164,184
,145,168,198,184,2682
B7 500 DATA 198,183,16,242,166,
181,164,184,202,189,65,3,145
,168,202,136,2444
F4 510 DATA 16,247,169,0,133,18
1,166,182,240,17,189,234,159
,157,65,3,2158
63 520 DATA 202,16,247,165,182,
133,181,169,0,133,182,76,68,
24,96,56,1930
EC 530 DATA 173,58,31,229,181,1
68,177,168,201,32,240,3,136,
208,247,200,2452
3F 540 DATA 162,0,177,168,157,2
34,0,0,0,0,0,0,0,0,0,898
0 TO 15:READ A:CX=CX+A:POKE
SA+L*16+D,A:NEXT D
A5 30 READ A:IF A>CX THENPRINT
"ERROR IN LINE";LN+(L*10):ST
OP
40 40 NEXT L:END
7E 50 DATA 159,169,32,145,168,2
32,200,204,57,31,208,240,134
,182,56,173,2390
9C 60 DATA 58,31,229,181,133,18
3,76,34,25,165,168,72,165,16
9,72,165,1926
ED 70 DATA 91,72,32,224,46,104,
133,91,104,133,169,104,133,1
68,56,165,1825
DB 80 DATA 168,237,57,31,133,16
8,176,2,198,169,32,20,30,198
,91,32,1742
DA 90 DATA 186,45,24,165,168,10
9,57,31,133,168,144,2,230,16
9,230,91,1952
4C 100 DATA 162,0,160,0,189,65,
3,145,168,200,232,228,181,20
8,245,32,2218
41 110 DATA 186,45,96,165,166,1
33,168,165,167,133,169,56,16
5,168,229,172,2383
2B 120 DATA 133,168,176,2,198,1
69,172,58,31,177,168,201,32,
208,32,132,2057
10 130 DATA 184,136,132,183,164
,183,177,168,164,184,145,168
,198,184,198,183,2751
FD 140 DATA 164,184,196,172,208
,238,169,32,145,168,169,148,
32,210,255,76,2566
91 150 DATA 120,31,127,127,127,
127,127,127,127,127,127,127,
127,127,127,127,1929
C4 160 DATA 127,127,127,127,127
,127,127,127,127,127,127,127
,127,127,127,2032
CA 170 DATA 127,127,127,127,127
,127,127,127,127,127,127,127
,127,127,127,2032
1C 180 DATA 127,127,127,127,127
,127,127,127,127,127,127,201
,1,208,6,32,1845
7B 190 DATA 227,46,76,120,31,20
1,2,208,3,76,106,49,201,3,20
8,11,1568
D9 200 DATA 173,68,31,208,3,32,
99,22,76,120,31,201,4,208,11
,173,1460
4F 210 DATA 68,31,208,244,32,56
,24,76,120,31,201,5,208,3,76
,196,1579
2C 220 DATA 47,201,6,208,3,76,8
2,50,201,7,208,3,76,90,35,20
1,1494
AC 230 DATA 8,208,3,76,93,35,20
1,9,208,3,76,246,27,201,10,2
08,1612
A3 240 DATA 3,76,121,27,201,11,
208,3,76,79,37,201,12,208,3,
76,1342
2B 250 DATA 71,52,201,13,208,3,
76,4,61,201,14,208,3,76,89,2
8,1308
89 260 DATA 201,15,208,3,76,99,
35,201,16,208,3,76,184,36,20
1,18,1580
AB 270 DATA 208,4,169,1,208,6,2
01,19,208,7,169,0,133,255,76
,74,1738
CE 280 DATA 52,201,20,208,3,76,
97,47,201,21,208,6,32,94,51,
76,1393

```

PROGRAM: YC WRITER D

```

14 10 BL=49 :LN=50 :SA=6521
5B 20 FOR L=0 TO BL:CX=0:FOR D=

```



```

03 290 DATA 120,31,201,23,240,2
4,201,24,208,3,76,182,28,201
,25,208,1795
67 300 DATA 3,76,185,28,201,26,
208,3,76,0,35,76,120,31,173,
66,1307
68 310 DATA 31,240,47,169,0,141
,66,31,141,67,31,32,60,40,16
9,0,1265
33 320 DATA 141,171,2,169,184,1
41,167,2,169,224,141,168,2,1
69,23,141,2014
6F 330 DATA 169,2,169,204,141,1
70,2,169,32,32,72,40,32,63,4
0,76,1413
8A 340 DATA 126,27,169,1,141,66
,31,133,177,32,60,40,169,0,1
41,171,1484
4C 350 DATA 2,169,184,141,167,2
,169,224,141,168,2,169,23,14
1,169,2,1873
99 360 DATA 169,204,141,170,2,1
69,215,32,72,40,32,63,40,76,
120,31,1576
D0 370 DATA 173,67,31,240,44,16
9,0,141,67,31,32,60,40,169,0
,141,1405
9B 380 DATA 171,2,169,200,141,1
67,2,169,224,141,168,2,169,2
5,141,169,2060
AD 390 DATA 2,169,204,141,170,2
,169,32,32,72,40,32,63,40,76
,120,1364
A0 400 DATA 31,169,1,141,67,31,
141,66,31,133,177,169,0,141,
171,2,1471
4B 410 DATA 169,184,141,167,2,1
69,224,141,168,2,169,23,141,
169,2,169,2040
9B 420 DATA 204,141,170,2,169,2
15,32,72,40,169,200,141,167,
2,169,224,2117
35 430 DATA 141,168,2,169,25,14
1,169,2,169,204,141,170,2,16
9,0,141,1813
32 440 DATA 171,2,169,202,32,72
,40,32,63,40,76,120,31,173,6
8,31,1322
EE 450 DATA 240,47,32,56,24,169
,0,141,68,31,32,60,40,169,0,
141,1250
4F 460 DATA 171,2,169,216,141,1
67,2,169,224,141,168,2,169,2
7,141,169,2078
7D 470 DATA 2,169,204,141,170,2
,169,32,32,72,40,32,63,40,76
,120,1364
9E 480 DATA 31,169,1,141,68,31,
32,60,40,169,0,141,171,2,169
,216,1441
DB 490 DATA 141,167,2,169,224,1
41,168,2,169,27,141,169,2,16
9,204,141,2036
26 500 DATA 170,2,169,201,32,72
,40,32,63,40,32,99,22,76,120
,31,1201
8B 510 DATA 32,60,40,169,104,14
1,167,2,169,225,141,168,2,16
9,45,141,1775
97 520 DATA 169,2,169,204,141,1
70,2,169,0,141,171,2,169,80,
133,178,1900
9F 530 DATA 169,31,133,179,169,
16,133,211,169,165,141,20,3,
169,194,141,2043
96 540 DATA 21,3,88,32,95,193,1
20,32,63,40,76,120,31,0,0,0,
914

```

## PROGRAM: YC WRITER E

```

01 10 BL=50 :LN=50 :SA=7350
5B 20 FOR L=0 TO BL:CX=0:FOR D=
0 TO 15:READ A:CX=CX+A:POKE
SA+L*16+D,A:NEXT D
AS 30 READ A:IF A>CX THENPRINT
"ERROR IN LINE":LN+(L*10):ST
OP
40 40 NEXT L:END
45 50 DATA 76,194,28,76,35,29,7
6,90,29,76,177,29,32,67,29,2
40,1283
52 60 DATA 3,76,120,31,165,172,
240,249,74,176,246,24,173,63
,31,208,2051
0F 70 DATA 240,101,172,133,172,
56,173,57,31,229,172,141,57,
31,168,136,2069
FE 80 DATA 140,58,31,165,172,14
1,63,31,74,133,73,170,24,173
,59,31,1538
54 90 DATA 105,8,141,59,31,144,
3,238,60,31,238,61,31,208,3,
238,1599
93 100 DATA 62,31,202,208,231,5
6,165,166,229,172,133,166,17
6,2,198,167,2364
7A 110 DATA 169,0,133,172,32,23
9,44,32,90,29,76,120,31,32,6
7,29,1295
09 120 DATA 240,3,76,120,31,165
,172,201,79,240,247,74,144,2
44,164,172,2372
SE 130 DATA 140,58,31,200,140,5
7,31,32,177,29,76,120,31,56,
173,64,1415
94 140 DATA 31,237,69,31,133,73
,173,65,31,237,70,31,5,73,24
0,1,1500
8B 150 DATA 96,165,171,96,173,1
67,2,133,168,133,87,173,168,
2,133,169,2036
C0 160 DATA 133,88,166,73,56,16
5,168,233,8,133,168,133,87,1
76,4,198,1989
0B 170 DATA 169,198,88,202,208,
238,169,22,133,74,166,73,169
,170,160,0,2239
EB 180 DATA 145,87,200,192,8,20
8,249,24,165,87,105,8,133,87
,144,2,1844
90 190 DATA 230,88,202,208,231,
24,165,168,105,64,133,168,13
3,87,165,169,2340
9E 200 DATA 105,1,133,169,133,8
8,198,74,208,208,96,173,167,
2,133,168,2056
2B 210 DATA 133,87,173,168,2,13
3,169,133,88,24,165,168,105,
8,133,168,1857
3C 220 DATA 133,87,144,4,230,16
9,230,88,169,22,133,73,56,16
9,80,229,2016
B1 230 DATA 172,237,63,31,74,17
0,169,85,160,0,145,87,200,19
2,8,208,2001
60 240 DATA 249,24,165,87,105,8
,133,87,144,2,230,88,202,208
,231,24,1987
29 250 DATA 165,168,105,64,133,
168,133,87,165,169,105,1,133
,169,133,88,1986
2B 260 DATA 198,73,208,200,96,1
27,127,127,127,127,127,127,1
27,127,172,58,2148

```

```

76 270 DATA 31,177,168,201,32,2
40,1,96,162,0,134,185,134,18
7,232,134,2114
49 280 DATA 186,160,0,177,168,2
01,32,208,13,177,168,201,32,
208,7,200,2138
C3 290 DATA 204,57,31,208,244,9
6,177,168,201,32,208,2,230,1
85,200,204,2447
BE 300 DATA 57,31,208,242,165,1
85,240,207,172,58,31,132,184
,136,177,168,2393
4B 310 DATA 201,32,208,5,230,18
6,136,208,245,132,183,56,165
,185,229,186,2587
BE 320 DATA 133,185,197,186,176
,22,162,1,56,165,186,229,185
,133,186,197,2399
A2 330 DATA 185,144,3,232,208,2
42,134,187,166,185,208,2,166
,186,164,183,2595
B6 340 DATA 177,168,164,184,145
,168,201,32,240,7,198,184,19
8,183,76,132,2457
02 350 DATA 30,198,184,164,184,
169,32,145,168,198,184,198,1
83,165,187,240,2629
0B 360 DATA 29,134,73,166,187,1
65,186,240,5,198,186,76,183,
30,202,240,2300
F2 370 DATA 11,164,184,169,32,1
45,168,198,184,202,208,245,1
66,73,202,208,2559
CC 380 DATA 189,96,255,255,255,
255,255,255,255,255,255,255,
255,255,255,255,3855
AB 390 DATA 255,255,255,255,255
,255,255,255,255,255,255,255
,255,255,255,255,4080
3E 400 DATA 255,255,255,255,255
,255,255,255,255,255,255,255
,255,255,255,255,4080
41 410 DATA 255,255,255,255,255
,255,255,255,255,255,127,127
,127,127,127,127,3312
C9 420 DATA 127,127,127,127,127
,127,127,127,127,127,127,127
,127,127,127,127,2032
4F 430 DATA 127,127,127,127,127
,127,127,127,127,127,127,127
,127,127,127,127,2032
9B 440 DATA 127,127,127,127,127
,127,127,127,127,127,127,12,
0,0,0,41,1450
3B 450 DATA 0,122,8,80,79,192,2
27,120,204,0,160,8,1,1,0,188
,1390
F2 460 DATA 62,155,159,127,8,12
7,127,64,83,58,78,79,32,78,6
5,77,1379
3B 470 DATA 69,32,32,32,32,32,3
2,32,32,32,127,127,127,127,1
27,127,1119
1B 480 DATA 127,127,127,127,127
,127,127,127,127,127,127,127
,76,129,31,76,1836
2E 490 DATA 226,31,76,14,32,76,
61,32,76,70,32,169,254,141,3
3,208,1531
7B 500 DATA 169,6,141,134,2,141
,32,208,169,128,141,138,2,16
0,7,169,1747
E4 510 DATA 0,153,193,2,136,16,
250,32,125,194,32,125,38,173
,69,31,1569
BA 520 DATA 133,166,133,73,141,
64,31,173,70,31,133,167,133,
74,141,65,1728

```

```

5C 530 DATA 31,32,78,40,32,81,4
    0,160,80,140,57,31,136,140,5
    8,31,1167
08 540 DATA 169,192,141,59,31,1
    69,227,141,60,31,169,120,141
    ,61,31,169,1911
38 550 DATA 204,0,0,0,0,0,0,0,0
    ,0,0,0,0,0,0,0,204

```

## PROGRAM: YC WRITER F

```

A7 10 BL=50 :LN=50 :SA=8151
SB 20 FOR L=0 TO BL: CX=0: FOR D=
    0 TO 15: READ A: CX=CX+A: POKE
    SA+L*16+D,A: NEXT D
AS 30 READ A: IF A<CX THEN PRINT
    "ERROR IN LINE"; LN+(L*10): ST
    OP
40 40 NEXT L: END
ED 50 DATA 141,62,31,169,0,141,
    63,31,76,14,32,173,55,31,133
    ,166,1318
3C 60 DATA 173,56,31,133,167,17
    3,54,31,133,171,173,53,31,13
    3,172,173,1857
FB 70 DATA 51,31,133,175,173,52
    ,31,133,176,173,49,31,133,17
    3,173,50,1737
3A 80 DATA 31,133,174,120,32,13
    2,194,32,228,255,240,251,201
    ,133,208,3,2367
45 90 DATA 76,246,54,201,135,20
    8,3,76,86,44,201,136,240,24,
    201,139,2070
0C 100 DATA 208,3,76,89,44,172,
    141,2,192,4,208,3,76,84,26,2
    01,1529
8D 110 DATA 134,240,212,76,138,
    32,32,68,229,32,152,195,32,1
    82,195,169,2118
D1 120 DATA 49,141,20,3,169,234
    ,141,21,3,169,202,141,38,3,1
    69,241,1744
68 130 DATA 141,39,3,165,166,14
    1,55,31,165,167,141,56,31,16
    5,171,141,1778
76 140 DATA 54,31,165,172,141,5
    3,31,165,175,141,51,31,165,1
    76,141,52,1744
5A 150 DATA 31,165,173,141,49,3
    1,165,174,141,50,31,96,255,2
    55,255,255,2267
AB 160 DATA 255,253,255,76,144,
    32,76,61,33,201,13,208,3,76,
    94,34,1814
51 170 DATA 201,17,208,3,76,211
    ,33,201,19,240,45,201,20,208
    ,3,76,1762
9C 180 DATA 119,46,201,32,208,3
    ,76,200,50,201,145,208,3,76,
    37,34,1639
99 190 DATA 201,148,208,3,76,22
    0,25,201,157,208,3,76,38,33,
    201,134,1932
69 200 DATA 208,3,76,123,31,76,
    254,41,120,173,59,31,141,167
    ,2,173,1678
84 210 DATA 60,31,141,168,2,173
    ,61,31,141,169,2,173,62,31,1
    41,170,1556
CF 220 DATA 2,169,0,141,171,2,8

```

```

8,56,165,166,229,172,133,166
    ,176,2,1838
C2 230 DATA 198,167,166,171,240
    ,15,56,165,166,237,57,31,133
    ,166,176,2,2146
49 240 DATA 198,167,202,208,241
    ,56,165,175,229,171,133,175,
    176,2,198,176,2672
FD 250 DATA 169,0,133,171,133,1
    72,32,236,44,32,239,44,76,12
    0,31,164,1796
8D 260 DATA 175,208,11,164,176,
    208,7,164,172,208,3,76,120,3
    1,164,166,2053
68 270 DATA 208,2,198,167,198,1
    66,198,172,164,172,192,255,2
    40,3,76,202,2613
77 280 DATA 33,165,171,208,37,3
    2,66,40,32,216,39,169,0,133,
    91,165,1597
4F 290 DATA 166,133,168,165,167
    ,133,169,56,165,168,237,58,3
    1,133,168,176,2293
9C 300 DATA 2,198,169,32,186,45
    ,120,76,147,33,198,171,120,5
    6,173,167,1893
88 310 DATA 2,233,64,141,167,2,
    173,168,2,233,1,141,168,2,56
    ,173,1726
76 320 DATA 169,2,233,40,141,16
    9,2,176,3,206,170,2,173,57,3
    1,74,1648
3F 330 DATA 170,24,173,167,2,10
    5,8,141,167,2,144,3,238,168,
    2,238,1752
7C 340 DATA 169,2,208,3,238,170
    ,2,202,208,231,88,169,157,32
    ,210,255,2344
27 350 DATA 173,58,31,133,172,1
    65,175,208,2,198,176,198,175
    ,32,236,44,2176
88 360 DATA 76,205,33,32,210,25
    5,32,239,44,76,120,31,173,72
    ,31,133,1762
AB 370 DATA 74,56,173,71,31,237
    ,57,31,133,73,176,2,198,74,5
    6,165,1607
06 380 DATA 73,229,166,133,75,1
    65,74,229,167,5,75,144,46,16
    0,0,177,1918
A3 390 DATA 166,133,73,24,165,1
    66,109,57,31,133,166,144,2,2
    30,167,165,1931
D9 400 DATA 171,201,21,208,6,32
    ,199,34,76,25,34,230,171,169
    ,17,32,1626
88 410 DATA 210,255,230,175,208
    ,2,230,176,32,236,44,76,120,
    31,165,175,2365
48 420 DATA 208,7,165,176,208,3
    ,76,120,31,160,0,177,166,133
    ,73,56,1759
75 430 DATA 165,166,237,57,31,1
    33,166,176,2,198,167,165,171
    ,208,6,32,2080
5C 440 DATA 217,34,76,83,34,198
    ,171,169,145,32,210,255,165,
    175,208,2,2174
6D 450 DATA 198,176,198,175,76,
    31,34,173,72,31,133,74,56,17
    3,71,31,1702
3E 460 DATA 237,57,31,133,73,17
    6,2,198,74,56,165,73,229,166
    ,133,75,1878
CB 470 DATA 165,74,229,167,5,75
    ,144,69,56,173,57,31,229,172
    ,24,101,1771

```

```

7B 480 DATA 166,133,166,144,2,2
    30,167,165,171,201,21,208,27
    ,165,172,72,2210
98 490 DATA 169,0,133,172,32,19
    9,34,104,170,240,18,120,169,
    157,32,210,1959
6F 500 DATA 255,202,208,250,88,
    76,180,34,230,171,32,1,42,23
    0,175,208,2382
A0 510 DATA 2,230,176,169,0,133
    ,172,32,239,44,32,236,44,76,
    120,31,1736
4D 520 DATA 32,60,40,32,66,40,3
    2,116,39,32,63,40,169,21,133
    ,91,1006
50 530 DATA 208,16,32,60,40,32,
    66,40,32,216,39,32,63,40,169
    ,0,1085
8A 540 DATA 133,91,165,167,133,
    169,165,166,133,168,56,165,1
    68,229,172,133,2413
C7 550 DATA 168,0,0,0,0,0,0,0,0
    ,0,0,0,0,0,0,0,168

```

## PROGRAM: YC WRITER G

```

6E 10 BL=52 :LN=50 :SA=8952
SB 20 FOR L=0 TO BL: CX=0: FOR D=
    0 TO 15: READ A: CX=CX+A: POKE
    SA+L*16+D,A: NEXT D
AS 30 READ A: IF A<CX THEN PRINT
    "ERROR IN LINE"; LN+(L*10): ST
    OP
40 40 NEXT L: END
75 50 DATA 176,2,198,169,32,186
    ,45,96,8,72,138,72,152,72,21
    6,166,1800
57 60 DATA 93,56,165,87,229,89,
    165,88,229,90,144,24,160,0,2
    32,196,2047
6F 70 DATA 92,208,3,202,240,46,
    177,87,145,89,200,208,242,23
    0,88,230,2487
36 80 DATA 90,76,23,35,164,92,1
    38,101,88,133,88,138,24,101,
    90,133,1514
85 90 DATA 90,232,152,240,8,136
    ,177,87,145,89,76,58,35,198,
    88,198,2009
21 100 DATA 90,202,208,241,104,
    168,104,170,104,40,96,127,12
    7,127,127,127,2162
FD 110 DATA 127,127,76,102,35,7
    6,5,36,76,230,35,76,32,36,17
    3,237,1479
FB 120 DATA 3,208,52,169,1,141,
    237,3,165,167,141,236,3,56,1
    65,166,1913
92 130 DATA 229,172,141,235,3,1
    76,3,206,236,3,32,60,40,32,1
    21,36,1725
6E 140 DATA 32,147,36,169,92,16
    0,36,32,30,171,166,175,165,1
    76,32,205,1824
2D 150 DATA 189,32,63,40,76,120
    ,31,201,2,240,67,169,2,141,2
    37,3,1613
61 160 DATA 165,167,141,234,3,5
    6,173,58,31,229,172,24,101,1
    66,141,233,2094
1A 170 DATA 3,144,3,238,234,3,3

```





```

2,141,62,3,173,170,2,141,63,
3,1309
C7 370 DATA 173,171,2,141,64,3,
96,120,169,165,141,20,3,169,
194,141,1772
1B 380 DATA 21,3,173,60,3,141,1
67,2,173,61,3,141,168,2,173,
62,1353
31 390 DATA 3,141,169,2,173,63,
3,141,170,2,173,64,3,141,171
,2,1421
8D 400 DATA 88,96,120,169,165,1
41,20,3,169,194,141,21,3,173
,60,3,1566
4D 410 DATA 141,167,2,133,251,1
73,61,3,141,168,2,133,252,17
3,62,3,1865
43 420 DATA 141,169,2,173,63,3,
141,170,2,173,64,3,141,171,2
,32,1450
CB 430 DATA 214,195,160,7,177,2
51,153,193,2,136,16,248,32,2
22,195,88,2289
2D 440 DATA 96,169,1,141,192,2,
173,167,2,141,190,2,133,251,
173,168,2001
B7 450 DATA 2,141,191,2,133,252
,160,7,185,193,2,145,251,136
,16,248,2064
C9 460 DATA 169,33,141,184,2,16
9,0,141,192,2,96,165,251,141
,188,2,1876
2E 470 DATA 165,252,141,189,2,1
73,167,2,141,190,2,133,251,1
73,168,2,2151
45 480 DATA 141,191,2,133,252,3
2,214,195,160,7,177,251,153,
193,2,174,2277
73 490 DATA 171,2,208,5,9,240,7
6,72,41,9,15,145,251,136,16,
234,1630
5F 500 DATA 32,222,195,96,169,1
,141,192,2,173,186,2,240,30,
169,0,1850
17 510 DATA 141,186,2,173,190,2
,133,251,173,191,2,133,252,1
60,7,185,2181
8D 520 DATA 193,2,145,251,136,1
6,248,169,33,141,184,2,169,0
,141,192,2022
5F 530 DATA 2,96,201,128,144,5,
56,233,128,208,3,56,233,32,1
41,172,1838
04 540 DATA 2,152,72,32,88,196,
104,168,96,169,192,133,251,1
69,227,32,2083
3D 550 DATA 236,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,236

```

PROGRAM: YC WRITER I

```

2E 10 BL=50 :LN=50 :SA=1065
4
5B 20 FOR L=0 TO BL:CX=0:FOR D=
0 TO 15:READ A:CX=CX+A:POKE
SA+L*16+D,A:NEXT D
AS 30 READ A:IF A>CX THENPRINT

```

```

"ERROR IN LINE";LN+(L*10):ST
OP
40 40 NEXT L:END
12 50 DATA 195,169,120,133,251,
169,204,32,16,196,160,7,169,
0,153,193,2167
33 60 DATA 2,136,16,250,96,56,1
73,71,31,237,69,31,72,173,72
,31,1516
09 70 DATA 237,70,31,170,160,0,
169,32,145,73,200,208,251,23
0,74,202,2252
CF 80 DATA 208,246,104,240,9,17
0,169,32,145,73,200,202,208,
250,96,160,2512
39 90 DATA 0,132,171,132,91,132
,175,132,176,132,172,132,173
,132,174,140,2196
34 100 DATA 68,31,140,237,3,200
,140,66,31,140,67,31,132,177
,96,255,1814
E1 110 DATA 76,4,42,76,232,43,1
33,75,165,166,205,71,31,208,
10,165,1702
CC 120 DATA 167,205,72,31,208,3
,76,120,31,165,75,201,29,240
,26,172,1821
1B 130 DATA 68,31,240,3,32,62,4
8,164,172,208,8,172,66,31,24
0,3,1548
E0 140 DATA 76,189,42,160,0,165
,75,145,166,230,166,208,2,23
0,167,56,2077
D1 150 DATA 165,166,237,64,31,1
33,77,165,167,237,65,31,5,77
,144,16,1780
36 160 DATA 165,75,201,29,240,1
0,165,166,141,64,31,165,167,
141,65,31,1856
6E 170 DATA 230,172,165,172,205
,57,31,208,75,169,0,133,177,
165,171,201,2331
6E 180 DATA 21,208,36,165,75,20
1,29,240,6,32,72,40,76,128,4
2,32,1403
8B 190 DATA 66,40,32,116,39,165
,166,133,168,165,167,133,169
,169,21,133,1882
0E 200 DATA 91,32,186,45,76,156
,42,230,171,165,75,32,210,25
5,32,232,2030
F7 210 DATA 43,169,0,133,172,32
,239,44,230,175,208,2,230,17
6,32,236,2121
E6 220 DATA 44,76,120,31,165,75
,32,210,255,32,239,44,76,120
,31,172,1722
02 230 DATA 68,31,240,7,160,1,1
32,177,76,49,42,164,175,208,
4,164,1698
13 240 DATA 176,240,15,201,32,2
08,14,164,177,208,7,160,1,13
2,177,76,1988
6D 250 DATA 120,31,76,49,42,133
,179,164,177,240,3,76,49,42,
32,236,1649
B2 260 DATA 44,165,167,133,169,
56,165,166,229,172,133,168,1
76,2,198,169,2312
F2 270 DATA 56,165,168,237,57,3
1,133,168,176,2,198,169,160,
0,177,168,2065
59 280 DATA 201,32,208,3,200,20
8,247,177,168,201,32,240,9,2
00,204,57,2387
15 290 DATA 31,208,244,76,49,42
,165,166,133,89,165,167,133,
90,165,89,2012

```

```

6D 300 DATA 208,2,198,90,198,89
,162,0,160,0,177,89,201,32,2
40,11,1857
E8 310 DATA 165,89,208,2,198,90
,198,89,232,208,239,134,178,
32,232,43,2337
E2 320 DATA 165,167,133,181,56,
165,166,229,178,133,180,176,
2,198,181,160,2470
F1 330 DATA 0,166,178,240,17,17
7,180,145,166,169,32,145,180
,169,29,32,2025
67 340 DATA 210,255,200,202,208
,239,230,178,165,179,145,166
,169,29,32,210,2817
FC 350 DATA 255,24,165,166,101,
178,133,166,144,2,230,167,24
,165,172,101,2193
BF 360 DATA 178,133,172,32,239,
44,56,165,166,237,64,31,133,
77,165,167,2059
CC 370 DATA 237,65,31,5,77,144,
10,165,166,141,64,31,165,167
,141,65,1674
34 380 DATA 31,169,1,133,177,16
5,167,133,169,56,165,166,229
,172,133,168,2234
A7 390 DATA 176,2,198,169,166,1
71,134,91,32,186,45,56,165,1
68,237,57,2053
6B 400 DATA 31,133,168,176,2,19
8,169,173,67,31,240,3,32,20,
30,166,1639
A3 410 DATA 171,202,134,91,32,1
86,45,76,120,31,120,169,0,14
1,171,2,1691
6B 420 DATA 173,59,31,141,167,2
,173,60,31,141,168,2,173,61,
31,141,1554
B1 430 DATA 169,2,173,62,31,141
,170,2,166,171,240,149,24,17
3,167,2,1842
9F 440 DATA 105,64,141,167,2,17
3,168,2,105,1,141,168,2,24,1
73,169,1605
0B 450 DATA 2,105,40,141,169,2,
144,3,238,170,2,202,208,222,
88,96,1832
F1 460 DATA 127,127,127,127,127
,127,127,127,127,127,127,127
,127,127,127,127,2032
F7 470 DATA 127,127,127,127,127
,127,127,127,127,127,127,127
,127,127,127,127,2032
E3 480 DATA 127,127,127,127,127
,127,127,127,76,92,44,76,150
,44,24,173,1695
C2 490 DATA 58,31,109,63,31,133
,73,24,165,172,109,63,31,170
,189,0,1421
9B 500 DATA 50,201,46,240,2,208
,11,228,73,240,23,189,0,50,2
01,46,1808
73 510 DATA 208,16,169,29,32,21
0,255,230,166,208,2,230,167,
230,172,232,2556
DC 520 DATA 208,229,32,239,44,7
6,120,31,24,165,172,109,63,3
1,170,236,1949
3C 530 DATA 63,31,240,238,189,0
,50,201,46,240,2,208,12,236,
63,31,1850
CS 540 DATA 240,224,189,0,50,20
1,46,208,217,169,157,32,210,
255,165,166,2529
7C 550 DATA 208,2,198,167,198,1
66,198,172,202,208,226,240,1
97,0,0,0,2382

```



## PROGRAM: YC WRITER J

```

12 10 BL=51 :LN=50 :SA=1150
5B 20 FOR L=0 TO BL:CX=0:FOR D=
0 TO 15:READ A:CX=CX+A:POKE
SA+L*16+D,A:NEXT D
A5 30 READ A:IF A>CX THENPRINT
"ERROR IN LINE";LN+(L*10):ST
OP
40 40 NEXT L:END
79 50 DATA 76,242,44,76,61,45,3
2,60,40,169,24,141,167,2,169
,224,1572
97 60 DATA 141,168,2,169,3,141,
169,2,169,204,141,170,2,169,
0,141,1791
8A 70 DATA 171,2,166,175,232,20
8,7,24,165,176,105,1,208,2,1
65,176,1983
81 80 DATA 32,205,189,165,176,2
08,22,165,175,201,10,176,7,1
69,32,32,1964
DB 90 DATA 210,255,208,6,201,10
0,176,5,169,32,32,210,255,32
,63,40,1994
FE 100 DATA 96,32,60,40,169,72,
141,167,2,169,224,141,168,2,
169,9,1661
52 110 DATA 141,169,2,169,204,1
41,170,2,169,0,141,171,2,166
,172,232,2051
88 120 DATA 169,0,32,205,189,16
5,172,201,10,176,5,169,32,32
,210,255,2022
FA 130 DATA 32,63,40,96,127,127
,127,127,127,127,127,127,127
,127,127,127,1755
7B 140 DATA 127,127,127,127,253
,255,255,255,255,255,255,255
,255,255,253,255,3564
3C 150 DATA 255,255,253,255,255
,255,255,255,255,255,255,255
,255,255,253,255,4076
86 160 DATA 255,255,255,255,255
,255,255,253,255,255,255,255
,253,255,255,253,4074
46 170 DATA 253,255,253,255,255
,255,255,255,76,189,45,76,21
3,45,76,21,2777
28 180 DATA 46,32,186,45,24,165
,168,109,57,31,133,168,144,2
,230,169,1709
F1 190 DATA 230,91,165,91,201,2
2,208,233,96,169,192,141,167
,2,169,227,2404
84 200 DATA 141,168,2,169,120,1
41,169,2,169,204,141,170,2,1
66,91,240,2095
48 210 DATA 34,24,173,167,2,105
,64,141,167,2,173,168,2,105,
1,141,1469
63 220 DATA 168,2,24,173,169,2,
105,40,141,169,2,144,3,238,1
70,2,1552
FC 230 DATA 202,208,222,169,0,1
41,171,2,96,165,91,201,0,144
,74,201,2087
3C 240 DATA 22,176,70,32,60,40,
32,183,45,173,63,31,240,39,1
73,57,1436
69 250 DATA 31,201,80,240,32,12
0,173,63,31,74,170,24,173,16
7,2,105,1686
E6 260 DATA 8,141,167,2,144,3,2
38,168,2,238,169,2,208,3,238

```

```

,170,1901
85 270 DATA 2,202,208,231,88,16
0,0,177,168,132,73,32,72,40,
164,73,1822
D3 280 DATA 200,204,57,31,208,2
41,32,84,40,96,127,127,127,1
27,127,127,1955
88 290 DATA 127,127,127,127,127
,127,127,127,127,127,164
,175,208,11,164,2119
02 300 DATA 176,208,7,164,172,2
08,3,76,120,31,56,173,57,31,
229,172,1883
37 310 DATA 133,178,165,166,133
,87,133,89,133,168,165,167,1
33,88,133,90,2161
91 320 DATA 133,169,165,89,208,
2,198,90,198,89,160,0,177,16
6,145,89,2078
43 330 DATA 200,196,178,208,247
,169,32,145,89,56,165,168,22
9,172,133,168,2555
12 340 DATA 176,2,198,169,165,1
71,133,91,32,186,45,165,166,
208,2,198,2107
C8 350 DATA 167,198,166,169,157
,76,141,32,255,255,255,255,2
55,255,255,255,3146
80 360 DATA 255,255,255,255,76,
230,46,76,56,47,173,65,31,13
3,90,133,2176
56 370 DATA 88,24,173,64,31,133
,87,109,57,31,133,89,144,2,2
30,90,1485
82 380 DATA 165,89,141,64,31,16
5,90,141,65,31,160,0,177,87,
145,89,1640
68 390 DATA 165,87,208,2,198,88
,198,87,165,89,208,2,198,90,
198,89,2072
20 400 DATA 165,90,197,169,208,
230,165,89,197,168,208,224,1
60,0,169,32,2471
80 410 DATA 145,168,200,204,57,
31,208,248,32,180,45,96,165,
167,133,169,2248
A4 420 DATA 56,165,166,229,172,
133,168,176,2,198,169,56,165
,168,237,64,2324
47 430 DATA 31,133,251,165,169,
237,65,31,5,251,144,1,96,166
,171,134,2050
8C 440 DATA 91,32,230,46,96,32,
60,40,169,128,141,167,2,169,
226,141,1770
4F 450 DATA 168,2,169,80,141,16
9,2,169,204,141,170,2,24,165
,172,109,1887
21 460 DATA 63,31,133,73,106,17
6,4,160,0,240,2,160,1,140,17
1,2,1462
1C 470 DATA 170,240,27,24,173,1
67,2,105,8,141,167,2,144,3,2
38,168,1779
CA 480 DATA 2,238,169,2,208,3,2
38,170,2,202,208,231,166,73,
189,0,2101
DB 490 DATA 50,201,46,240,5,169
,46,76,184,47,169,47,157,0,5
0,32,1519
81 500 DATA 210,255,32,63,40,76
,120,31,32,123,31,32,68,229,
24,162,1528
E7 510 DATA 12,160,4,32,240,255
,169,247,160,47,32,30,171,16
9,9,133,1870
84 520 DATA 211,169,24,160,48,3
2,30,171,32,228,255,240,251,

```

201,78,240,2370

```

01 530 DATA 7,201,89,208,243,76
,114,31,76,117,31,82,69,65,6
8,89,1566
24 540 DATA 32,84,79,32,69,82,6
5,83,69,32,87,72,79,76,69,32
,1042
DD 550 DATA 84,69,88,84,70,73,7
6,69,33,13,13,0,18,32,65,82,
869
43 560 DATA 69,32,89,79,85,32,8
3,85,82,69,0,0,0,0,0,0,705

```

## PROGRAM: YC WRITER K

```

31 10 BL=48 :LN=50 :SA=1232
6
5B 20 FOR L=0 TO BL:CX=0:FOR D=
0 TO 15:READ A:CX=CX+A:POKE
SA+L*16+D,A:NEXT D
A5 30 READ A:IF A>CX THENPRINT
"ERROR IN LINE";LN+(L*10):ST
OP
40 40 NEXT L:END
7C 50 DATA 63,32,40,89,47,78,41
,32,146,0,127,127,127,127,12
7,127,1330
73 60 DATA 127,127,127,127,127,
127,127,127,165,166,133,93,1
33,168,165,167,2206
05 70 DATA 133,94,133,169,165,9
4,133,98,56,165,93,229,172,1
33,97,133,2097
7C 80 DATA 168,176,4,198,98,198
,169,165,171,133,91,133,80,1
33,86,76,2079
AD 90 DATA 115,48,24,165,97,105
,2,133,97,144,2,230,98,206,5
8,31,1555
8E 100 DATA 24,165,97,109,58,31
,133,97,144,2,230,98,238,58,
31,230,1745
0B 110 DATA 91,230,86,160,0,177
,97,201,32,208,9,200,177,97,
201,32,1998
23 120 DATA 208,208,240,71,160,
2,177,97,201,32,208,198,200,
177,97,201,2477
0E 130 DATA 32,208,191,177,97,2
01,32,208,14,200,204,57,31,2
08,244,230,2334
4B 140 DATA 97,208,40,230,98,20
8,36,165,91,72,165,168,72,16
5,169,72,2056
C0 150 DATA 165,98,133,169,24,1
65,97,105,2,133,168,144,2,23
0,169,32,1836
DE 160 DATA 224,46,104,133,169,
104,133,168,104,133,91,230,9
7,208,2,230,2176
1B 170 DATA 98,56,165,97,229,16
6,133,99,165,98,229,167,133,
100,56,165,2156
DB 180 DATA 97,237,64,31,133,73
,165,98,237,65,31,5,73,144,1
0,165,1628
2C 190 DATA 97,141,64,31,165,98
,141,65,31,165,97,133,95,165
,98,133,1719
A0 200 DATA 96,165,95,208,2,198
,96,198,95,165,93,197,97,208
,6,165,2084
2C 210 DATA 94,197,98,240,25,16
0,0,177,95,145,97,165,95,208
,2,198,1996

```

```

94 220 DATA 96,198,95,165,97,20
    8,2,198,98,198,97,76,31,49,1
    65,80,1853
A1 230 DATA 133,91,32,186,45,24
    ,165,168,109,57,31,133,168,1
    44,2,230,1718
2E 240 DATA 169,230,80,165,80,1
    97,86,208,229,96,127,127,127
    ,127,127,127,2302
2C 250 DATA 127,127,127,127,165
    ,167,133,169,133,88,133,90,5
    6,165,166,229,2202
74 260 DATA 172,133,168,133,89,
    176,4,198,169,198,90,24,165,
    168,109,57,2053
07 270 DATA 31,133,87,144,2,230
    ,88,56,165,87,237,64,31,133,
    91,165,1744
DE 280 DATA 88,237,65,31,5,91,1
    44,3,76,120,31,160,0,177,87,
    145,1460
EB 290 DATA 89,230,89,208,2,230
    ,90,230,87,208,2,230,88,173,
    64,31,2051
7A 300 DATA 197,87,208,231,173,
    65,31,197,88,208,224,160,0,1
    65,87,197,2318
3E 310 DATA 89,208,6,165,88,197
    ,90,240,14,165,87,208,2,198,
    88,198,2043
63 320 DATA 87,169,32,145,87,20
    8,230,165,171,133,91,32,180,
    45,165,89,2029
04 330 DATA 141,64,31,165,90,14
    1,65,31,76,120,31,253,255,25
    5,255,255,2228
E5 340 DATA 255,255,255,255,255
    ,255,255,255,255,255,221,46,
    46,46,46,58,3013
2F 350 DATA 46,46,46,46,221,46,
    46,46,46,58,46,46,46,46,221,
    46,1098
EE 360 DATA 46,46,46,58,46,46,4
    6,46,221,46,46,46,46,58,46,4
    6,935
57 370 DATA 46,46,221,46,46,46,
    46,58,46,46,46,46,221,46,46,
    46,1098
32 380 DATA 46,58,46,46,46,46,2
    21,46,46,46,46,58,46,46,46,4
    6,935
9E 390 DATA 221,46,46,46,46,58,
    46,46,46,46,0,127,32,60,40,1
    69,1075
2F 400 DATA 128,141,167,2,169,2
    26,141,168,2,169,80,141,169,
    2,169,204,2078
99 410 DATA 141,170,2,169,0,141
    ,171,2,162,0,189,0,50,201,46
    ,240,1684
78 420 DATA 27,201,47,240,23,16
    9,46,157,0,50,134,74,32,72,4
    0,166,1478
28 430 DATA 74,232,224,80,208,2
    39,32,63,40,76,120,31,162,79
    ,189,19,1868
84 440 DATA 39,157,0,50,202,16,
    247,162,0,189,0,50,134,73,32
    ,72,1423
87 450 DATA 40,166,73,232,224,8
    0,208,241,76,140,50,255,255,
    255,255,255,2805
BC 460 DATA 255,255,255,255,255
    ,255,255,255,255,255,255,253
    ,255,255,8,40,3616
D0 470 DATA 8,40,76,206,50,76,5
    ,51,56,165,166,237,64,31,133
    ,73,1437

```

```

5B 480 DATA 165,167,237,65,31,5
    ,73,176,2,144,31,165,166,208
    ,2,198,1835
F9 490 DATA 167,198,166,160,0,1
    77,166,201,32,240,9,230,173,
    208,2,230,2359
44 500 DATA 174,32,5,51,230,166
    ,208,2,230,167,169,32,76,254
    ,41,32,1869
1F 510 DATA 60,40,169,112,141,1
    67,2,169,224,141,168,2,169,1
    3,141,169,1887
7B 520 DATA 2,169,204,141,170,2
    ,169,1,141,171,2,166,173,165
    ,174,32,1882
86 530 DATA 205,189,165,174,208
    ,22,165,0,0,0,0,0,0,0,0,11
    28

```

## PROGRAM: YC WRITER L

```

2C 10 BL=50 :LN=50 :SA=1310
    1
5B 20 FOR L=0 TO BL:CX=0:FOR D=
    0 TO 15:READ A:CX=CX+A:POKE
    SA+L*16+D,A:NEXT D
AS 30 READ A:IF A>CX THENPRINT
    "ERROR IN LINE":LN+(L*10):ST
    OP
40 40 NEXT L:END
00 50 DATA 173,201,10,176,7,169
    ,32,32,210,255,208,6,201,100
    ,176,5,1961
26 60 DATA 169,32,32,210,255,32
    ,63,40,96,77,79,86,69,32,84,
    65,1421
EE 70 DATA 66,80,79,83,73,84,73
    ,79,78,32,70,79,82,87,65,82,
    1192
38 80 DATA 68,169,0,133,173,133
    ,174,165,166,133,87,165,167,
    133,88,230,2184
0F 90 DATA 87,208,2,230,88,165,
    87,205,64,31,208,8,165,88,20
    5,65,1906
F4 100 DATA 31,208,1,96,120,173
    ,69,31,133,87,133,92,173,70,
    31,133,1581
EA 110 DATA 88,133,93,24,165,92
    ,109,58,31,133,92,144,2,230,
    93,169,1656
C6 120 DATA 0,133,89,133,90,133
    ,91,160,0,177,87,201,32,240,
    6,169,1741
63 130 DATA 0,133,89,240,14,165
    ,89,208,10,169,1,133,89,230,
    173,208,1951
47 140 DATA 2,230,174,230,87,20
    8,2,230,88,165,87,205,64,31,
    208,215,2226
3C 150 DATA 165,88,205,65,31,20
    8,208,56,173,64,31,229,92,13
    3,94,173,2015
68 160 DATA 65,31,229,93,5,94,1
    44,35,160,0,177,92,201,32,24
    0,13,1611
81 170 DATA 200,177,92,201,32,2
    40,6,230,90,208,2,230,91,24,
    165,92,2080
00 180 DATA 109,57,31,133,92,14
    4,208,230,93,176,204,24,165,
    173,101,90,2030

```

```

5B 190 DATA 133,173,165,174,101
    ,91,133,174,88,32,29,52,32,2
    03,50,96,1726
7C 200 DATA 32,60,40,169,112,14
    1,167,2,169,224,141,168,2,16
    9,13,141,1750
85 210 DATA 169,2,169,204,141,1
    70,2,169,1,141,171,2,162,5,1
    69,32,1709
E2 220 DATA 32,210,255,202,208,
    250,32,63,40,96,76,77,52,76,
    132,53,1854
88 230 DATA 32,123,31,32,92,193
    ,169,60,133,178,169,3,133,17
    9,162,15,1704
EB 240 DATA 189,97,31,201,32,20
    8,3,202,208,246,232,138,162,
    97,160,31,2237
00 250 DATA 32,189,255,169,1,17
    4,74,31,168,32,186,255,169,0
    ,32,213,1980
49 260 DATA 255,144,13,173,74,3
    1,201,1,240,3,32,170,54,76,1
    17,31,1615
06 270 DATA 142,64,31,140,65,31
    ,169,0,133,198,173,69,31,133
    ,168,133,1680
29 280 DATA 166,173,70,31,133,1
    69,133,167,133,252,56,165,16
    6,233,7,133,2187
27 290 DATA 251,176,2,198,252,1
    60,0,177,251,141,57,31,200,1
    77,251,141,2465
12 300 DATA 58,31,200,177,251,1
    41,59,31,200,177,251,141,60,
    31,200,177,2185
47 310 DATA 251,141,61,31,200,1
    77,251,141,62,31,200,177,251
    ,141,63,31,2209
20 320 DATA 32,81,40,173,59,31,
    141,167,2,173,60,31,141,168,
    2,173,1474
DA 330 DATA 61,31,141,169,2,173
    ,62,31,141,170,2,169,0,141,1
    71,2,1466
CE 340 DATA 32,180,45,173,63,31
    ,208,13,173,57,31,201,80,240
    ,68,32,1627
98 350 DATA 60,40,76,41,53,32,6
    0,40,173,63,31,74,133,73,32,
    188,1169
C9 360 DATA 28,24,173,63,31,109
    ,57,31,201,80,240,36,173,58,
    31,74,1409
6E 370 DATA 170,24,173,167,2,10
    5,8,141,167,2,144,3,238,168,
    2,202,1716
F5 380 DATA 208,239,165,172,72,
    173,58,31,133,172,32,191,28,
    104,133,172,2083
97 390 DATA 32,63,40,120,32,135
    ,194,32,236,44,32,239,44,32,
    94,51,1420
0C 400 DATA 76,120,31,18,32,80,
    82,69,83,83,32,65,78,89,32,7
    5,1045
C1 410 DATA 69,89,32,84,79,32,8
    2,69,84,85,82,78,32,84,79,32
    ,1092
16 420 DATA 84,69,88,84,32,146,
    0,32,123,31,32,68,229,24,162
    ,12,1216
1F 430 DATA 160,8,32,240,255,17
    3,74,31,201,1,240,10,165,255
    ,240,6,2091
0C 440 DATA 32,8,62,76,170,53,1
    69,80,160,54,32,30,171,169,9
    ,133,1408

```



```

E5 450 DATA 211,169,106,160,54,
    32,30,171,32,228,255,240,251
    ,201,78,208,2426
07 460 DATA 3,76,77,54,201,89,2
    08,240,173,74,31,201,1,240,7
    ,165,1840
84 470 DATA 255,240,3,32,11,62,
    173,70,31,133,252,56,173,69,
    31,233,1824
59 480 DATA 7,133,251,176,2,198
    ,252,160,0,173,57,31,145,251
    ,200,173,2209
AF 490 DATA 58,31,145,251,200,1
    73,59,31,145,251,200,173,60,
    31,145,251,2204
91 500 DATA 200,173,61,31,145,2
    51,200,173,62,31,145,251,200
    ,173,63,31,2190
11 510 DATA 145,251,169,60,133,
    178,169,3,133,179,162,16,202
    ,189,80,31,2100
0F 520 DATA 201,32,240,248,232,
    138,162,80,160,31,32,189,255
    ,169,3,174,2346
60 530 DATA 74,31,160,255,32,18
    6,255,169,251,174,64,31,172,
    65,31,232,2182
7B 540 DATA 208,1,200,32,221,24
    5,173,74,31,201,1,240,3,32,1
    70,54,1886
BB 550 DATA 0,0,0,0,0,0,0,0,0,0
    ,0,0,0,0,0,0,0,0,0,0

```

PROGRAM: YC WRITER M

```

14 10 BL=50 :LN=50 :SA=1390
    1
5B 20 FOR L=0 TO BL:CX=0:FOR D=
    0 TO 15:READ A:CX=CX+A:POKE
    SA+L*16+D,A:NEXT D
AS 30 READ A:IF A>CX THENPRINT
    "ERROR IN LINE";LN+(L*10):ST
    OP
40 40 NEXT L:END
04 50 DATA 76,117,31,82,69,65,6
    8,89,32,84,79,32,83,65,86,69
    ,1127
5D 60 DATA 32,84,69,88,84,70,73
    ,76,69,33,13,13,0,18,32,65,8
    19
3D 70 DATA 82,69,32,89,79,85,32
    ,83,85,82,69,63,32,40,89,47,
    1058
99 80 DATA 78,41,32,146,13,13,0
    ,169,0,32,189,255,169,15,162
    ,8,1322
1E 90 DATA 168,32,186,255,32,19
    2,255,162,15,32,198,255,166,
    144,208,9,2309
D3 100 DATA 32,207,255,32,210,2
    55,76,153,54,32,204,255,96,1
    69,13,32,2075
F3 110 DATA 210,255,32,210,255,
    169,7,133,211,169,1,133,204,
    32,132,54,2207
49 120 DATA 169,13,32,210,255,3
    2,210,255,169,4,133,211,169,
    96,160,53,2171
DB 130 DATA 32,30,171,32,228,25
    5,240,251,96,169,13,32,210,2
    55,32,210,2256

```

```

FB 140 DATA 255,169,4,133,211,1
    69,110,160,53,32,30,171,32,2
    28,255,240,2252
FB 150 DATA 251,96,81,173,33,20
    8,133,82,173,32,68,229,32,15
    2,195,32,1970
4B 160 DATA 182,195,173,32,208,
    133,81,173,33,208,133,82,173
    ,134,2,133,2075
CD 170 DATA 83,169,0,141,32,208
    ,141,33,208,169,1,133,80,32,
    170,55,1655
85 180 DATA 32,228,255,240,251,
    201,13,208,24,165,80,208,10,
    169,1,133,2218
67 190 DATA 80,32,170,55,76,29,
    55,169,0,133,80,32,147,58,76
    ,29,1221
6F 200 DATA 55,201,133,208,219,
    165,81,141,32,208,165,82,141
    ,33,208,165,2237
EF 210 DATA 83,141,134,2,120,32
    ,132,194,76,120,31,32,68,229
    ,76,126,1596
41 220 DATA 31,7,171,60,67,79,7
    6,79,85,82,0,108,7,154,83,75
    ,1164
F3 230 DATA 73,80,53,0,118,7,17
    9,62,67,79,76,79,85,82,0,128
    ,1168
F7 240 DATA 7,59,0,138,7,83,75,
    73,80,53,168,35,60,84,69,88,
    1079
DF 250 DATA 84,50,0,148,7,177,3
    5,62,84,69,88,84,50,0,158,7,
    1103
EF 260 DATA 153,36,65,66,49,69,
    0,168,7,59,0,178,7,76,179,55
    ,1167
E1 270 DATA 76,61,56,76,86,56,1
    69,0,133,87,169,4,133,88,169
    ,0,1363
93 280 DATA 133,89,169,216,133,
    90,32,68,229,160,0,169,85,14
    5,87,169,1974
A1 290 DATA 6,145,89,200,169,67
    ,145,87,169,6,145,89,192,38,
    208,243,1998
9B 300 DATA 200,169,73,145,87,1
    69,6,145,89,32,61,56,185,150
    ,56,240,1863
30 310 DATA 9,145,87,169,6,145,
    89,200,208,242,32,61,56,185,
    191,56,1881
76 320 DATA 240,9,145,87,169,6,
    145,89,200,208,242,32,61,56,
    169,74,1932
1E 330 DATA 145,87,169,6,145,89
    ,200,169,67,145,87,169,6,145
    ,89,192,1910
F2 340 DATA 38,208,243,200,169,
    75,145,87,169,6,145,89,32,61
    ,56,32,1755
13 350 DATA 61,56,169,232,133,8
    4,169,56,133,85,162,16,32,86
    ,56,96,1626
DE 360 DATA 24,165,87,105,40,13
    3,87,144,2,230,88,24,165,89,
    105,40,1528
30 370 DATA 133,89,144,2,230,90
    ,160,0,96,169,0,133,73,160,0
    ,177,1656
23 380 DATA 84,240,36,201,254,2
    40,15,201,255,208,17,32,61,5
    6,230,84,2214
5D 390 DATA 208,235,230,85,208,
    231,169,128,133,73,208,242,6
    9,73,145,87,2524

```

```

26 400 DATA 169,6,145,89,200,20
    8,216,24,200,152,101,84,133,
    84,144,2,1957
95 410 DATA 230,85,32,61,56,202
    ,208,197,96,93,32,32,32,32,3
    2,32,1452
69 420 DATA 32,32,32,42,42,42,3
    2,39,25,3,32,23,18,9,20,5,42
    8
C5 430 DATA 18,39,32,42,42,42,3
    2,32,32,32,32,32,32,32,32,3
    2,535
83 440 DATA 93,0,93,40,3,41,32,
    49,57,56,55,32,2,45,8,32,638
B0 450 DATA 12,5,8,13,1,14,14,3
    2,32,32,32,32,22,5,18,19,291
C0 460 DATA 9,15,14,32,1,47,53,
    47,49,93,0,20,5,24,20,45,474
C4 470 DATA 5,14,20,18,25,32,13
    ,15,4,5,19,58,0,45,45,45,363
57 480 DATA 45,45,45,45,45,45,4
    5,45,45,45,45,45,45,0,23,
    653
A9 490 DATA 32,45,32,23,15,18,4
    ,32,23,18,1,16,32,15,14,47,3
    67
BC 500 DATA 15,6,6,0,10,32,45,3
    2,18,9,7,8,20,32,10,21,271
C5 510 DATA 19,20,9,6,9,3,1,20,
    9,15,14,32,15,14,47,15,248
94 520 DATA 6,6,0,9,32,45,32,9,
    14,19,5,18,20,32,13,15,275
A6 530 DATA 4,5,32,15,14,47,15,
    6,6,0,255,6,15,18,13,1,452
63 540 DATA 20,9,14,7,32,6,1,3,
    9,12,9,20,9,5,19,58,233
BB 550 DATA 0,0,0,0,0,0,0,0,0,0
    ,0,0,0,0,0,0,0,0,0,0

```

PROGRAM: YC WRITER N

```

15 10 BL=50 :LN=50 :SA=1470
    1
5B 20 FOR L=0 TO BL:CX=0:FOR D=
    0 TO 15:READ A:CX=CX+A:POKE
    SA+L*16+D,A:NEXT D
AS 30 READ A:IF A>CX THENPRINT
    "ERROR IN LINE";LN+(L*10):ST
    OP
40 40 NEXT L:END
E6 50 DATA 0,45,45,45,45,45,45,
    45,45,45,45,45,45,45,45,6
    75
28 60 DATA 45,45,45,45,45,45,0,
    3,32,45,32,21,14,45,10,21,49
    3
8C 70 DATA 19,20,9,6,25,32,16,1
    ,18,1,7,18,1,16,8,32,229
C7 80 DATA 21,14,4,5,18,32,3,21
    ,18,19,15,18,0,4,32,45,269
8E 90 DATA 32,18,9,7,8,20,32,10
    ,21,19,20,9,6,25,32,16,284
9A 100 DATA 1,18,1,7,18,1,16,8,
    32,21,14,4,5,18,32,3,199
EC 110 DATA 21,18,19,15,18,0,1,
    32,45,32,9,14,19,5,18,20,286

```

```

38 120 DATA 32,5,13,16,20,25,32
    ,12,9,14,5,32,1,20,32,3,271
BC 130 DATA 21,18,19,15,18,0,25
    ,5,5,18,1,19,5,32,6,1,3,436
2D 140 DATA 9,12,9,20,9,5,19,58
    ,0,45,45,45,45,45,45,45,456
FE 150 DATA 45,45,45,45,45,45,4
    ,5,45,45,45,0,2,32,45,32,5,56
    6
DC 160 DATA 18,1,19,5,32,12,9,1
    ,4,5,32,21,14,4,5,18,32,241
D0 170 DATA 3,21,18,19,15,18,0,
    ,11,32,45,32,5,18,1,19,5,262
EC 180 DATA 32,3,21,18,18,5,14,
    ,20,32,2,12,15,3,11,0,5,211
0C 190 DATA 32,45,32,5,18,1,19,
    ,5,32,23,8,15,12,5,32,20,304
64 200 DATA 5,24,20,6,9,12,5,0,
    ,255,255,254,32,32,6,49,32,99
    6
09 210 DATA 45,32,2,1,3,11,32,2
    ,0,15,32,20,5,24,20,32,32,326
DE 220 DATA 32,18,5,20,32,45,32
    ,13,15,18,5,32,8,5,12,16,308
2A 230 DATA 32,32,32,0,89,0,169
    ,0,133,87,169,4,133,88,169,0
    ,1137
BF 240 DATA 133,89,169,216,133,
    ,90,32,68,229,169,180,133,84,
    ,169,58,133,2085
F7 250 DATA 85,162,21,32,176,55
    ,96,2,12,15,3,11,45,13,15,22
    ,765
D9 260 DATA 5,32,6,1,3,9,12,9,2
    ,0,9,5,19,58,0,45,45,278
FD 270 DATA 45,45,45,45,45,45,4
    ,5,45,45,45,45,45,45,45,45
    ,720
FC 280 DATA 45,45,45,45,0,7,32,
    ,45,32,19,5,20,32,19,20,1,412
0F 290 DATA 18,20,47,5,14,4,32,
    ,15,6,32,2,12,15,3,11,0,236
67 300 DATA 8,32,45,32,3,15,16,
    ,25,32,3,21,18,18,5,14,20,307
91 310 DATA 32,2,12,15,3,11,32,
    ,20,15,32,3,21,18,19,15,18,26
    8
CC 320 DATA 0,15,32,45,32,13,15
    ,22,5,32,3,21,18,18,5,14,290
2D 330 DATA 20,32,2,12,15,3,11,
    ,32,20,15,32,3,21,18,19,15,27
    0
CB 340 DATA 18,0,255,13,1,18,7,
    ,9,14,32,6,1,3,9,12,9,407
99 350 DATA 20,9,5,19,58,0,45,4
    ,5,45,45,45,45,45,45,45,56
    1
FD 360 DATA 45,45,45,45,45,45,4
    ,5,45,0,24,32,45,32,19,5,20,5
    37
E4 370 DATA 32,12,5,6,20,32,13,
    ,1,18,7,9,14,0,25,32,45,271
FE 380 DATA 32,19,5,20,32,18,9,
    ,7,8,20,32,13,1,18,7,9,250
C2 390 DATA 14,0,255,20,1,2,32,
    ,6,1,3,9,12,9,20,9,5,398
3C 400 DATA 19,58,0,45,45,45,45
    ,45,45,45,45,45,45,45,45,45
    ,662
D2 410 DATA 45,45,0,6,53,32,45,
    ,32,10,21,13,16,32,20,15,32,4
    17
B4 420 DATA 14,5,24,20,32,20,1,

```

```

    2,45,16,15,19,9,20,9,15,266
C3 430 DATA 14,0,6,54,32,45,32,
    ,10,21,13,16,32,20,15,32,6,34
    8
F1 440 DATA 15,18,13,5,18,32,20
    ,1,2,45,16,15,19,9,20,9,257
4D 450 DATA 15,14,0,6,32,45,32,
    ,3,12,5,1,18,47,18,5,19,272
48 460 DATA 20,15,18,5,32,1,12,
    ,12,32,20,1,2,45,16,15,19,265
EA 470 DATA 9,20,9,15,14,19,0,2
    ,0,32,45,32,3,12,5,1,18,254
F4 480 DATA 47,18,5,19,20,15,18
    ,5,32,19,9,14,7,12,5,32,277
12 490 DATA 20,1,2,45,16,15,19,
    ,9,20,9,15,14,0,255,15,20,475
A4 500 DATA 8,5,18,19,58,0,45,4
    ,5,45,45,45,45,45,32,32,32,51
    9
9A 510 DATA 32,32,32,32,32,32,3
    ,2,32,32,93,32,21,32,45,32,23
    ,566
4F 520 DATA 15,18,4,3,15,21,14,
    ,20,0,12,32,45,32,12,15,1,259
04 530 DATA 4,32,6,9,12,5,32,32
    ,32,32,32,32,93,32,14,32,431
F7 540 DATA 45,32,3,8,1,14,7,5,
    ,32,6,9,12,5,14,1,13,207
BB 550 DATA 0,0,0,0,0,0,0,0,0,0
    ,0,0,0,0,0,0,0,0,0,0,0,0

```

## PROGRAM: YC WRITER D

```

78 10 BL=30 :LN=50 :SA=1550
    1
5B 20 FOR L=0 TO BL:CX=0:FOR D=
    0 TO 15:READ A:CX=CX+A:POKE
    SA+L*16+D,A:NEXT D
A5 30 READ A:IF A>CX THENPRINT
    "ERROR IN LINE":LN+(L*10):ST
    OP
AC 40 NEXT L:PRINT"[DOWN2]GET R
    EADY TO SAVE AND PRESS A KEY
    ":POKE198,0:WAIT198,1
ED 45 POKE43,99:POKE44,22:POKE4
    5,142:POKE46,62:X=PEEK(186):
    SAVE"WRITER2".X
2F 50 DATA 5,0,19,32,45,32,19,1
    ,22,5,32,6,9,12,5,32,276
EE 60 DATA 32,32,32,32,32,93,32
    ,13,32,45,32,3,8,1,14,7,440
7C 70 DATA 5,32,3,15,12,15,21,1
    ,8,19,0,18,32,45,32,18,5,290
1F 80 DATA 16,12,1,3,5,32,6,9,1
    ,2,5,32,32,32,93,32,16,338
98 90 DATA 32,45,32,16,18,9,14,
    ,20,15,21,20,0,255,254,32,32,
    815
B2 100 DATA 6,49,32,45,32,2,1,3
    ,11,32,20,15,32,20,5,24,329
5B 110 DATA 20,32,32,32,18,5,20
    ,32,45,32,13,15,18,5,32,8,35
    9
BA 120 DATA 5,12,16,32,32,32,0,
    ,32,123,31,32,68,229,24,162,1
    0,840
4E 130 DATA 160,10,32,240,255,5
    ,6,173,32,208,233,240,133,77,
    ,169,104,160,2282
C7 140 DATA 61,32,169,61,144,3,
    ,141,32,208,56,173,33,208,233
    ,240,133,1927

```

```

51 150 DATA 77,169,120,160,61,3
    ,2,169,61,144,3,141,33,208,17
    ,3,134,2,1687
A3 160 DATA 133,77,169,135,160,
    ,61,32,169,61,144,3,141,134,2
    ,198,211,1830
16 170 DATA 169,148,160,61,32,3
    ,0,171,32,228,255,240,251,201
    ,78,240,170,2466
7D 180 DATA 201,89,208,243,120,
    ,32,132,194,76,120,31,66,79,8
    ,2,68,69,1810
AC 190 DATA 82,32,67,79,76,79,8
    ,5,82,58,32,0,80,65,80,69,82,
    1048
54 200 DATA 32,67,79,76,79,85,8
    ,2,58,32,0,73,78,75,32,67,79,
    994
04 210 DATA 76,79,85,82,58,32,0
    ,18,32,83,65,84,73,83,70,73,
    993
5A 220 DATA 69,68,63,32,40,89,4
    ,7,78,41,32,146,0,32,30,171,1
    65,1103
C3 230 DATA 211,72,169,0,166,77
    ,32,205,189,104,133,211,162,
    0,134,73,1938
AD 240 DATA 134,74,134,75,32,22
    ,8,255,240,251,201,13,240,26,
    201,48,144,2296
35 250 DATA 243,201,58,176,239,
    ,32,210,255,73,48,166,75,149,
    73,232,224,2454
3E 260 DATA 2,134,75,240,2,208,
    ,221,169,13,32,210,255,32,210
    ,255,169,2227
B5 270 DATA 10,133,211,166,75,2
    ,08,2,24,96,24,162,10,169,0,1
    01,73,1464
5E 280 DATA 202,208,251,101,74,
    ,56,96,59,42,42,42,76,14,62,7
    6,22,1423
22 290 DATA 62,169,88,160,62,32
    ,30,171,96,162,0,189,78,31,1
    57,65,1552
3D 300 DATA 3,201,32,208,11,232
    ,189,78,31,201,32,208,238,20
    2,208,5,2079
24 310 DATA 232,224,18,208,230,
    ,169,0,157,65,3,169,0,32,189,
    255,169,2120
8E 320 DATA 15,174,74,31,168,32
    ,186,255,32,192,255,162,15,3
    2,201,255,2079
1A 330 DATA 169,65,160,3,32,30,
    ,171,32,181,171,96,82,69,65,6
    8,89,1483
8B 340 DATA 32,84,79,32,82,69,8
    ,0,76,65,67,69,32,84,69,88,84
    ,1092
27 350 DATA 70,73,76,69,33,13,1
    ,3,0,0,0,0,0,0,0,0,0,0,347

```

## PROGRAM: WRITER BOOT

```

C9 5 A=A+1:Y=PEEK(186)
13 10 IFA=1THENLOAD"WRITER1",Y,
    1
B9 20 IFA=2THENLOAD"WRITER2",Y,
    1
43 30 IFA=3THENSYS8050

```



# Technical Information

*All you ever wanted to know about your Commodore but were afraid to ask.*

Most programmers spend a lot of their time sifting through piles of technical books looking for the address of a certain routine or trying to find the POKE to perform a certain function.

Now you can throw away your books, as on the following pages you will find a wealth of information about all of the popular Commodore computers.

Advanced programmers will find the memory maps

invaluable while both beginners and old hands alike will find the Hex converter, the hints and tips and much more, to their liking.

Most of the information provided here is useful by itself. Some information, such as the addresses of routines within the ROMs, will be of more use when used together with a ROM disassembly.

C16 OP-SYSTEM AND MEMORY MAP			
HEX ADDRESS	BIT	DESCRIPTION OF ROUTINE	
\$0001		Cassette control	
	4	Cassette read	
	3	Cassette motor (0-on)	
	1	Cassette write	
\$000A		Q-LOAD: 1-VERIFY	
\$000D		Type: FF=string, 00=numeric	
\$000E		Type: 00=integer, 00=floating	
\$000F		DATA scan/LIST quote/memory flag	
\$0014-0015		Integer value	
\$0016		Pointer temporary string stack	
\$0017-0018		Last temp string vector	
\$0019-0021		Temporary string stack	
\$0022-0025		Utility pointer area	
\$0026-002A		Product area for multiplication	
\$002B-002C		Pointer: start of basic	
\$002D-002E		Pointer: start of basic variables	
\$002F-0030		Pointer: start of arrays	
\$0031-0032		Pointer: end of arrays	
\$0033-0034		Pointer: bottom of strings	
\$0035-0036		Pointer: current string	
\$0037-0038		Pointer: top of basic memory	
\$0039-003A		Current basic line number	
\$003B-003C		CHARGEI pointer	
\$003D-003E		Pointer: CONT basic line number	
\$003F-0040		Current DATA line number	
\$0041-0042		Current DATA address	
\$0043-0044		Input vector	
\$0045-0046		Current variable name	
\$0047-0048		Current variable address	
\$0049-004A		Variable pointer for FOR/NEXT	
\$004B-004C		Y save/op save/basic pointer save	
\$004D		Comparison symbol accumulator	
\$004E-0053		Misc numeric work area	
\$0054-0056		Jump vector for functions	
\$0057-0058		Misc work area	
\$0059		FAC#1 exponent	
\$005A		FAC#1 mantissa	
\$005B		FAC#1 sign	
\$005C		Series evaluation constant pointer	
\$005D		FAC#1 overflow	
\$005E-005F		FAC#2	
\$0060		FAC sign comparison	
\$0061		FAC#1 rounding	
\$0062-0065		Room for graphics screen (0=not available)	
\$0066		Pointer: GOSUB stack	
\$0067		Flag for window (540=window on, 500=multicolor, 500=both)	
\$0068		Status word ST	
\$0069		Keyswitch CIA: STOP and RVS flags	
\$006A		Q-load: 1-verify	
\$006B		Serial output: deferred char flag	
\$006C		Serial deferred character	
\$006D		Number of open files	
\$006E		Input device	
\$006F		Output CMD device	
\$006A		Direct-80:run=0 output control	
\$006D-006E		Tape end address/end of program	
\$00A3-00A5		Jiffy clock	
\$00A6		Serial bit count/EOI flag	
\$00AA		Countdown tape write/bit count	
\$00AB		number characters in filename	
\$00AC		Current logical file	
\$00AD		Current secondary address	
\$00AE		Current device	
\$00AF-00B0		Pointer to filename	
\$00B2-00B3		I/O start address	
\$00B4-00B5		Alt start address (load/verify)	
\$00B6-00B7		Pointer: cassette buffer	
\$00C4-00C5		Input cursor log (row,column)	
\$00C6		which key (64=no key)	
\$00C7		Input from screen/keyboard	
\$00CB-00CC		Pointer to screen line	
\$00CA		Pointer: cursor column	
\$00CB		Output quotes flag	
\$00CD		Pointer: cursor row	
\$00CE		Output character (to screen)	
\$00CF		Number of inserts outstanding	
\$00EA-00EB		Screen color pointer	
\$00EC-00ED		Keyboard pointer	
\$00EF		Number of characters in keyboard buffer	
\$00FB		Type of tape file	
\$00FF-010A		Floating to ASCII work area	
\$0100-013E		Tape error log	
\$0100-01FF		Processor stack area	
\$0200-025B		Basic input buffer	
\$0259-025A		Pointer: line number for CONT	
\$025B-025C		Pointer: basic statement for CONT	
\$02F2-02F3		Float-fixed vector	
\$02F4-02F5		Fixed-Float vector	
\$0300-0311		Basic vectors	
\$0312-0313		IRQ vector for keyscan/clock	
\$0314-0315		Main IRQ vector for sound duration/graphics split	
\$0316-0317		BRK interrupt vector	
\$0318-0319		OPEN vector	
\$031A-031B		CLOSE vector	
\$031C-031D		Set input vector	
\$031E-031F		Set output vector	
\$0320-0321		Restore I/O vector	
\$0322-0323		INPUT vector	
\$0324-0325		OUTPUT vector	
\$0326-0327		Test STOP vector	
\$0328-0329		GET vector	
\$032A-032B		Abort I/O vector	
\$032C-032D		User vector	
\$032E-032F		LOAD vector	
\$0330-0331		SAVE vector	
\$0333-03F2		Cassette buffer	
\$0473		CHARGEI subroutine	
\$04FC/04FE		Duration for voice 1	
\$04FD/04FF		Duration for voice 2/noise	
\$0503		RND seed value	
\$0509-0512		Logical file table	
\$0513-051C		Device number table	
\$051D-0526		Secondary address table	
\$0527-0530		Keyboard buffer	
\$0531-0532		Start of usable memory	
\$0533-0534		End of usable memory	
\$0535		Serial bus timeout flag	

YOUR COMMODORE SERIOUS USERS GUIDE  
72



\$F445	Monitor call entry
\$F44C	Monitor BRK entry (\$0316) user vector (\$032C)
\$FC19	Get I/O address
\$FC83	IRQ entry
\$FC8E	IRQ exit
\$FD00	IED memory
\$FF52	Perform MONITOR

## \*4 MEMORY MAP

DEC ADDRESS	DESCRIPTION
0	7501 Data direction register
1	7501 8 bit I/O port (asC64)
3-4	New start address (RENUMBER)
5-6	Step width (RENUMBER)
7	Search character
8	Flag: searching for quote
9	Screen column from last TAB
10	Flag: 0=LOAD, 1=VERIFY
11	Input buffer counter, number of elements
13	Flag: \$FF=string, \$00=numeric
14	Flag: \$FF=integer, \$00=floating
15	Flag: Data scan/LIST quote/memory flag
16	Flag: user function call
17	Flag: \$00=INPUT, \$40=GET, \$90=READ
20-21	Integer value
22	Pointer: Temporary string stack
23-24	Vector: Last temporary string
25-33	Temporary string stack
34-37	Utility pointer area
38-42	Product area for multiplication
43-44	Pointer: start of basic
45-46	Pointer: start of basic variables
47-48	Pointer: start of basic arrays
49-50	Pointer: end of arrays
51-52	Pointer: start of strings
53-54	Pointer: current string
55-56	Pointer: top of basic memory
57-58	Current basic line number
59-60	Previous basic line number
61-62	Pointer: CONT basic line number
63-64	Current DATA line number
65-66	Pointer: current DATA address
67-68	Vector: input routine
69-70	Current variable name
71-72	Variable address
73-74	Variable pointer for FOR/NEXT
75-76	Y save, op save, basic pointer save
77	Comparison: 1=larger, 2=equal, 4=smaller
78-83	Misc work area
84-86	Vector: functions
87-96	Misc work area
97	Accumulator #1 exponent
98-101	Accumulator #1 mantissa
102	Accumulator #1 sign
103	Series evaluation constant pointer
104	Accumulator #1 hi-order overflow
105-110	Accumulator #2 as for #1
111	Accumulator sign comparison
113	Accumulator #1 rounding
113-114	Pointer: cassette buffer
115-116	Flag: AUTO command \$00=OFF
117	Flag: \$01= 10K reserved for graphics
124-125	Pointer: GOSUB stack
131	Current graphics mode: \$00=text, \$20= hires, \$50=split hires
132	\$A0=multicolor, \$E0=multicolor split
133	Current color
134	Multi-color 1
135	Foreground color
136	Max number of columns
137	Max number of rows
144	Status word ST
145	Flag: STOP and RVS keys
147	Flag: \$00=LOAD, \$01=VERIFY
148	Flag: character in serial buffer \$00=NO, \$80=YES
149	Character in buffer for serial address
151	Number of files open
152	Default input device
153	Device output device
154	Flag: \$80=direct mode, \$C0=monitor, \$00=program
157-158	Pointer: tape end/program end
163-165	Jiffy clock
171	Length of filename
172	Logical file number
173	Secondary address
174	Device number
175-176	Pointer: filename
178-179	I/O start address
180-181	Basic loading address
182-183	Pointer: load end address for tape
194	Flag: RVS \$12=Yes, \$00=No
196-197	Cursor position (x,y)
198	Flag: key pressed: \$40=none
199	Input from screen/keyboard
200-201	Pointer: screen line
202	Pointer: screen column
203	Flag: \$00=not in quote mode
204	Length of current screen line
205	Pointer: cursor row
206	Output character to screen
207	Flag: insert mode, >\$00=number of inserts
234-235	Pointer: current screen color
236-238	Vector to keyboard decode
239	Number of characters in keyboard buffer
248	Type of tape file
249	Bit 7=1:WRITE, Bit 6=1:READ
275-289	Color luminance table in RAM
291-311	Processor stack
312-600	Basic input buffer
601-602	Previous basic line number
603-604	Pointer: Basic statement for CONT
754-755	Pointer: Float to fixed routine
756-757	Pointer: fixed to integer
768-769	Vector: basic error messages
770-771	Vector: basic warm start

772-773	Vector: basic token generator
774-775	Vector: basic LIST
776-777	Vector: basic command execute
778-779	Vector: basic token evaluate
780-781	Vector: basic user token evaluate
782-783	Vector: create keyword
784-785	Vector: prepare user token
786-787	Vector: interrupt
788-789	Vector: hardware interrupt
790-791	Vector: BRK interrupt
792-793	Vector: kernel OPEN
794-795	Vector: CLOSE
796-797	Vector: CHKIN
798-799	Vector: CHKOUT
800-801	Vector: CLRCHN
802-803	Vector: CHRIN
804-805	Vector: CHROUT
806-807	Vector: STOP
808-809	Vector: GETIN
810-811	Vector: CLALL
812-813	Vector: monitor break
814-815	Vector: LOAD
816-817	Vector: SAVE
818-1010	Tape buffer
1139-1144	CHARGET subroutine
1145-1156	CHRGOT subroutine
1263	Last error number
1264-1265	Row number of last error
1266-1267	Reference for ON ERROR GOTO
1280	USR jump command
1281-1282	USR address (lo/hi)
1283	RND seed value
1289-1298	Table of logical file numbers
1299-1308	Table of device numbers
1309-1318	Table of secondary addresses
1319-1328	Keyboard buffer
1329-1330	Start address of RAM for OS
1331-1332	Pointer: end of RAM OS
1339	Current color code:
	BIT 7:1=Flash
	BIT 6:4=luminance (0-7)
	BIT 3:0:color (0-15)
1343	Size of keyboard buffer
1344	Flag: Key repeat: \$80=All, \$40=None, \$00=DEL, SPACE, CURSORS
1345	Repeat speed
1346	Repeat delay counter
1347	Flag: shift, ctrl, cbm key
1348	Last pattern of shift
1349-1350	Pointer: keyboard table setup
1351	Flag: SHIFT: \$80=NO, \$00=YES
1362	Program counter hi
1363	Program counter lo
1364	Processor flags
1365	Processor A reg
1366	Processor X reg
1367	Processor Y reg
1368	Processor stack pointer
2038	Current key pressed

## LOW MEMORY MAP OF C64

LABEL	HEX	DECIMAL	DESCRIPTION
DB510	\$0000	0	6510 Direction register
RB510	\$0001	1	6510 I/O, Memory and tape
ADRAY1	\$0003-0004	3-4	Float to fixed vector
ADRAY2	\$0005-0006	5-6	Fixed to float vector
CHARAC	\$0007	7	Search character
ENDCHR	\$0008	8	End of quote flag
TRMPOS	\$0009	9	Save screen last TAB
VERCK	\$000A	10	Flag: LOAD=0 VERIFY=1
COUNT	\$000B	11	Pointer input buffer/#subscripts
DIMFLG	\$000C	12	Default DIM to 10 flag
VALTYP	\$000D	13	Data type: String=255 Numeric=0
			Integer=128
			Float=0
INTFLG	\$000E	14	DATA scan/LIST quote/Garbage collect
GARBFL	\$000F	15	Flag
SUBFLG	\$0010	16	Subscript/User Fn call
INPFLG	\$0011	17	Input flag: \$00=Input \$40=Get \$80=Read
TANFLG	\$0012	18	TAN sign/comparison
	\$0013	19	Current I/O prompt
LINNUM	\$0014-0015	20-21	Integer value
TEMPPT	\$0016	22	Pointer temp string stack
LASTPT	\$0017-0018	23-24	Last temp string address
TEMPST	\$0019-0021	25-33	Stack for temp strings
INDEX	\$0022-0025	34-37	Utility pointer area
RESHO	\$0026-002A	38-42	Product area for multiply
TXTAB	\$002B-002C	43-44	Pointer start of BASIC
VRTAB	\$002D-002E	45-46	Pointer start of variables
ARYTAB	\$002F-0030	47-48	Pointer start of arrays
STREND	\$0031-0032	49-50	Pointer end of arrays
FRETOP	\$0033-0034	51-52	Pointer bottom of strings
FRESPEC	\$0035-0036	53-54	Utility string pointer
MEMSIZ	\$0037-0038	55-56	Pointer highest address used by BASIC
CURLIN	\$0039-003A	57-58	Current BASIC line number
OLDLIN	\$003B-003C	59-60	Previous BASIC line number
OLDTIX	\$003D-003E	61-62	BASIC statement for CONT
DATLIN	\$003F-0040	63-64	Current DATA line
DATPTR	\$0041-0042	65-66	Current DATA address
INPPTR	\$0043-0044	67-68	INPUT vector
VARNAM	\$0045-0046	69-70	Pointer current variable name
VARPNT	\$0047-0048	71-72	Pointer current variable data
FORPNT	\$0049-004A	73-74	Pointer variable for FOR/NEXT
	\$004B-004C	75-76	Y-save/op-save/BASIC pointer save
	\$004D	77	Comparison symbol accumulator
	\$004E-0053	78-83	Misc work area
	\$0054-0056	84-86	Jump vector for functions
	\$0057-0060	87-96	Misc numeric work area
FACEXP	\$0061	97	FPACC#1 - exponent
FACHO	\$0062-0065	98-101	FPACC#1 - mantissa
FACSGN	\$0066	102	FPACC#1 - sign
SGNFLG	\$0067	103	Pointer series evaluation constant
BITS	\$0068	104	FPACC#1 - overflow digit
ARGEXP	\$0069	105	FPACC#2 - exponent
ARGHO	\$006A-006D	106-109	FPACC#2 - mantissa
ARGSGN	\$006E	110	FPACC#2 - sign

▶

74



\$0050-0051	80-81	Var ptr for FN defin., + for garb coll.
\$0052-0054	82-84	Pntr: descriptor var list-string compares
\$0055	85	Help flag: \$xx=HELP, \$xx=LIST
\$0056-0057	86-87	Jump vector for function evaluations
\$0058	88	Oldov
\$0059	89	Area for INSTRING oper./temp pointer 1
\$005A-005B	90-91	Pointer: block transfer, DIM init
\$005C-005D	92-93	Pointer: block transfer
\$005E	94	Temp pnt 2, occasionally floating-pt acc
\$005F-0060	95-96	# places before/after dec. for conver.
\$0061	97	Pntr: Dec. pt when reading digit strings
\$0062	98	Exponent sign of the # read (neg.= \$80)
\$0063	99	Floating-pt. accumulator 1: Exponent
\$0064-0067	100-103	Floating-pt. accumulator 1: Mantissa
\$0068	104	Floating-pt. accumulator 1: sign
\$0069	105	pointer: Polynomial evaluation
\$006A	106	Floating-pt. accumulator 2: Exponent
\$006B-006E	107-110	Floating-pt. accumulator 2: Mantissa
\$006F	111	Floating-pt. acc. 2: sign
\$0070	112	Result flag: sign compare Acc 1 to Acc 2
\$0071	113	Floating-pt. accumulator 1: Round off
\$0072-0073	114-115	Pointer: Cassette buffer
\$0074-0075	116-117	Offset value for AUTO command, \$00=off
\$0076	118	Hires flag: 1=BASIC-start set 10k higher
\$0077	119	Sprite number-counter for leading zeros
\$0078	120	Help counter
\$0079	121	Temp storage for indirect loading
\$007A-007C	122-124	Description of error-variable D\$
\$007D-007E	125-126	End-of-stack during program run
\$007F	127	Mode flag: \$xx=RUN mode, \$xx=direct mode
\$0080	128	USING pnt for dec pnt., Stat.DOS parser
\$0081	129	Paratx
\$0082	130	Oldstx
\$0083	131	Current colour for graphic mode
\$0084	132	Multi-colour Mode: colour 1
\$0085	133	Multi-colour Mode: colour 2
\$0086	134	Foreground colour
\$0087-0088	135-136	X-direction scale factor
\$0089-008A	137-138	Y-direction scale factor
\$008B	139	Stop drawing, if not background colour
\$008C-008D	140-141	Address pointer for graphic routines
\$008E	142	Temp storage 1 for graphic routines
\$008F	143	Temp storage 2 for graphic routines
\$0090	144	Status word for kernel input/output
\$0091	145	Stop flag: STOP key, RVS key
\$0092	146	Time constants for cassette operations
\$0093	147	Load flag: \$00=LOAD, \$01=VERIFY
\$0094	148	Serial bus flag: character in buffer
\$0095	149	Char. in buffer for serial bus
\$0096	150	Sync # for cass, EOI received from tape
\$0097	151	Temporary data address
\$0098	152	Index for file tables, no. of open files
\$0099	153	Standard input device (0 for keyboard)
\$009A	154	Standard output device (3 for screen)
\$009B	155	Parity byte from cassette
\$009C	156	Tape flag: byte received
\$009D	157	Status flag for kernel
\$009E	158	Cassette error pass 1: char error
\$009F	159	Cassette error pass 2: corrected
\$00A0-00A2	160-162	24-hr real-time clock: 1/60-sec count
\$00A3-00A4	163-164	Temporary storage for serial bus
\$00A5	165	Countdown - SAVE on tape, ser. help ptr.
\$00A6	166	Pointer for cassette buffer
\$00A7	167	Tape short counter, RS-232 input bits
\$00A8	168	Tape read err, RS-232 counter input bits
\$00A9	169	Load 0 read flag, RS-232 start bit flag
\$00AA	170	Tape READ mode, RS-232 buffer input byte
\$00AB	171	Tape short counter, RS-232 input parity
\$00AC-00AD	172-173	Pointer: screen scroll, cass buffer Lo/Hi
\$00AE-00AF	174-175	pointer: program end, cassette end Lo/Hi
\$00B0-00B1	176-177	Cassette constant for time
\$00B2-00B3	178-179	Pointer: Start of cassette buffer Lo/Hi
\$00B4	180	Tape help pnt, RS232 next bit for scroll
\$00B5	181	EOI chart, RS-232 next bit for transfer
\$00B6	182	Tape help pointer, RS-232 byte buffer
\$00B7	183	Length of current filename
\$00B8	184	Logical file number (LFN)
\$00B9	185	Current secondary address (SA)
\$00BA	186	Current device number (GA)
\$00BB-00BC	187-188	Pntr: Address of current filename Lo/Hi
\$00BD	189	Tape pnt, RS-232 rotate parity buffer
\$00BE	190	No. of remaining read/write blocks
\$00BF	191	Serial buffer
\$00C0	192	Flag: cassette motor
\$00C1	193	Start address in/output (Lo), track no.
\$00C2	194	Start address in/output (Hi), sector no.
\$00C3-00C4	195-196	Tape LOAD temp. pnt kernel vector address
\$00C5	197	Tape read/write data range
\$00C6	198	Bank no. current LOAD, SAVE, VERIFY calls
\$00C7	199	Bank no. of current filename \$BB, \$BC
\$00C8-00C9	200-201	Pointer: RS-232 input buffer
\$00CA-00CB	202-203	Pointer: RS-232 output buffer
\$00CC-00CD	204-205	Pointer: keyboard decoder table
\$00CE-00CF	206-207	Pntr to string pos.-kernel PRINT routine
\$00D0	208	Index to keyboard buffer queue
\$00D1	209	Function key call flag
\$00D2	210	Function key string call index
\$00D3	211	Shift flag: Shift=\$01, C=\$02, Ctrl=\$04, old=\$08
\$00D4	212	Flag for keypress
\$00D5	213	Flag current pressed key (CHRS(0)=none)
\$00D6	214	Flag for INPUT or GET -- keyboard input
\$00D7	215	Flag for 40/80 column mode
\$00D8	216	Flag for text/graphic screen mode
\$00D9	217	Pointer for char set, RAM/ROM (only bit 2)
\$00DA	218	Pointer for MOULIN (Lo), <keysiz, bitmask>
\$00DB	219	Pointer for MOULIN (Hi), <keylen, saver>
\$00DC	220	Number of the function key
\$00DD	221	F-key string length up to current F-key
\$00DE	222	Bank for function key call (sedtl)
\$00DF	223	F-key string length up to current (F-key-1)
\$00E0-00E1	224-225	Pointer to running screen line: text RAM
\$00E2-00E3	226-227	pointer running screen line: attribute RAM
\$00E4	228	Lower border of window
\$00E5	229	Upper border of window
\$00E6	230	Left border of window
\$00E7	231	Right border of window
\$00E8	232	Start of running input column
\$00E9	233	Start of running input line
\$00EA	234	End of running input line
\$00EB	235	Current cursor position: line
\$00EC	236	Current cursor position: column
\$00ED	237	Maximum number of screen lines

\$00EE	238	Maximum number of screen columns
\$00EF	239	Temp storage of characters to be put out
\$00F0	240	Memory: previous char (for ESC test)
\$00F1	241	Colour code under cursor for char output
\$00F2	242	Colour code protection for INSERT/DELETE
\$00F3	243	Flag: RVS mode active
\$00F4	244	Flag: Quote mode active
\$00F5	245	Flag: Insert mode active
\$00F6	246	Flag: Auto insert active
\$00F7	247	Cutoff switching of C-Shift (\$B0) and Ctrl S (\$40)
\$00F8	248	Cutoff of screen scrolling
\$00F9	249	Cutoff of beep tones made by Ctrl G
\$00FA-00FE	250-254	Free area for user applications
\$00FF	255	LoFbuf

NOTE:- When using the C128 in C64 mode the Zero page mem map is the same as for the C64

*4 KERNAL JUMP TABLE				
LABEL	HEX ADD	DEC ADD	CODE	DESCRIPTION
	\$FF49	65353	JMP \$B7C2	Define function key no. after \$76, addr. after \$22-23, length in acc.
	\$FF4C	65356	JMP \$DC49	Print
	\$FF4F	65359	JMP \$F8D8	Print message
	\$FF52	65362	JMP \$F445	Call M/C Monitor
	\$FF55	65365	---	Not used
	\$FF7E	65406	---	Not used
	\$FF7F	65407	\$2A	Not used
	\$FF80	65408	\$B4	Not used
CINT	\$FF81	65409	JMP \$D84E	Initialise editor
IOINIT	\$FF84	65412	JMP \$F30B	Initialise I/O
RAMTAS	\$FF87	65415	JMP \$F352	Init RAM, open cass buffer, scn to \$C000
RESTOR	\$FF8A	65418	JMP \$F2CE	Restore vectors
VECTOR	\$FF8D	65421	JMP \$F2D3	Vector RAM
SETMSG	\$FF90	65424	JMP \$F41A	Control KERNAL messages
SECONO	\$FF93	65427	JMP \$EE4D	Send sec addr to listen
IKSA	\$FF96	65430	JMP \$EE1A	Send sec addr to talk
MENTOP	\$FF99	65433	JMP \$F427	set/read top RAM pointer
MEMBOT	\$FF9C	65436	JMP \$F436	set/read bottom RAM pointer
GCNKEY	\$FF9F	65439	JMP \$D811	Scan keyboard
SETTIMO	\$FFA2	65442	JMP \$F423	Set timeout for (optional) IEC-Bus
ACPIR	\$FFA5	65445	JMP \$EC8B	Input byte from serial port
CIOUT	\$FFA8	65448	JMP \$ECDF	Output byte via serial port
UNTLK	\$FFAB	65451	JMP \$EF3B	Command serial bus to UNTALK
UNLSN	\$FFAE	65454	JMP \$EF23	Command serial bus to UNLISTED
LISTEN	\$FFB1	65457	JMP \$EE2C	Command ALL devices on bus to listen
TALK	\$FFB4	65460	JMP \$E0FA	Command device on serial bus to talk
READST	\$FFB7	65463	JMP \$F41C	Read I/O status word
SETLFS	\$FFBA	65466	JMP \$F413	Set logical primary and secondary addr
SEINAM	\$FFBD	65469	JMP \$F40C	Determine file names
OPEN	\$FFC0	65472	JMP (\$031B) Open a logical file	\$EF53
CLOSE	\$FFC3	65475	JMP (\$031A) Close a logical file	\$EE5D
CHKIN	\$FFC6	65478	JMP (\$031C) Open channel for input	\$E018
CHKOUT	\$FFC9	65481	JMP (\$031E) Open channel for output	\$E060
CLRCHN	\$FFCC	65484	JMP (\$0320) Close I/O channels	\$EF0C
CHRIN	\$FFCF	65487	JMP (\$0322) Input character	\$E8E8
CHROUT	\$FFD2	65490	JMP (\$0324) Output character	\$EC48
LOAD	\$FFD5	65493	JMP \$F043	Load from peripheral device
SAVE	\$FFD8	65496	JMP \$F194	Store on peripheral device
SETTIM	\$FFDB	65499	JMP \$CF2D	Set time
ROTIM	\$FFDE	65502	JMP \$CF26	Read time
STOP	\$FFE1	65505	JMP (\$0326) Scan STOP key	\$F265
GETIN	\$FFE4	65508	JMP (\$032B) Read char from kb buffer	\$EBD9
CLALL	\$FFE7	65511	JMP (\$032A) Close all channels and logical files	
UDTIM	\$FFEA	65514	JMP \$CEFE	Increment time
SCREEN	\$FFED	65517	JMP \$D834	Identify X,Y screen set-up
PLOT	\$FFF0	65520	JMP \$D839	Read/set X,Y cursor positioning
IOBASE	\$FFF3	65523	JMP \$FC19	Base addr-reports back on I/O devices
	\$FFF6	65526	JMP \$FF3E	Switch on ROM
	\$FFF9	65529	JMP \$F2A4	Jump to reset routine
	\$FFFC	65532	JMP \$FFFE	Processor reset
	\$FFFE	65534	JMP \$FCR3	Processor interrupt

C64 KERNAL ROUTINES	
HEX ADDRESS	DESCRIPTION
\$E45F	Messages of the operating system
\$E4E0	Waits for Commodore key
\$E4EC	Constants for RS"K" timing
\$E500	Gets BASIC-address of the CIA or the VIA
\$E505	Gets screen format line/column
\$E50A	Set cursor or get cursor position
\$E518	Screen reset
\$E544	Clear screen
\$E566	Cursor home
\$E5A0	Initialize video controller
\$E5B4	Get character from keyboard buffer
\$E5CA	Waiting loop for keyboard input
\$E632	Get a character from the screen
\$E6B4	Checks for quote
\$E6B6	Calculate MSB for line starts
\$E6DA	Table of colour codes
\$E6EA	Scroll screen
\$E9CB	Shift line up
\$E9FF	Clear screen line
\$EA1C	Set character and colour on screen
\$EA24	Calculate pointer on colour RAM
\$EA31	Interrupt routine
\$EA87	Keyboard prompt
\$EB4B	Checks on shift, ctrl and commodore keys
\$EB79	Pointer on keyboard decoding tables
\$EB81	Decoding tables
\$EC44	Checks on control character
\$EC78	Decoding tables
\$EC89	Constants for video controller
\$ECE7	'Load (cr) Run (cr)'
\$ECF0	LSB tables of screen starts
\$ED09	Send TALK
\$ED0C	Send LISTEN
\$ED40	Output of byte on IEC-bus

\$E0B9	Send secondary address for LISTEN
\$E0C7	Send secondary address for TALK
\$E0EF	Send UNTALK
\$E0FE	Send UNLISTEN
\$EE13	Get a byte from the IEC-bus
\$EEB3	One millisecond delay
\$EEB8	Output RS232
\$EF4A	Calculate number of RS232 data-bits
\$F014	Output in RS232 buffer
\$F086	GET of RS232
\$F0A4	Set timer for IEC time-out
\$F0B0	Error messages of the operating system
\$F12B	Put out messages
\$F157	BASIN get a character
\$F1CA	BSOUT output a character
\$F20E	CHKIN Fixing of the input-device
\$F250	CKOUT Fixing of the output-device
\$F291	CLOSE
\$F30F	Look for logical file number
\$F31F	Set file parameter
\$F32F	CLALL closes all I/O channels
\$F34A	OPEN
\$F49E	LOAD
\$F5AF	Output 'Searching for file name'
\$F5D2	Output 'Loading/verifying'
\$F5D0	SAVE
\$F68F	Output 'Saving filename'
\$F69B	UDTIM increase running time
\$F6D0	Get time
\$F6E4	Set time
\$F6ED	Ask stop-key
\$F6FB	Put out error messages of the operating system
\$F72C	Read program header of tape
\$F76A	Write header on tape
\$F7D0	Get start address of tape buffer
\$F7D7	Set start and end address of the tape buffer
\$F7EA	Look for name on tape-header
\$F80D	Increase tape buffer pointer
\$F817	Waits for tape key for reading
\$F82E	Asks for tape key
\$F83B	Waits for tape key for writing
\$F841	Read block of tape
\$F84A	Load program of tape
\$F864	Write tape buffer on tape
\$F86B	Write block or program on tape
\$F8BE	Wait for I/O end
\$F8E1	Checks on stop key
\$F92C	Read interrupt routine for tape
\$F997	Set bit counter for serial output
\$F9A6	Write on bit on tape
\$F9CD	Write interrupt routine for tape
\$F9CB	Set IRQ vector
\$FCCA	Switch off tape drive
\$FCD1	Checks on reaching of end address
\$FCD8	Increase address pointer
\$FCE2	RESET
\$FD02	Checks on ROM in \$B000 or \$A000
\$FD10	ROM module identification
\$FD15	Set or get hardware and I/O vectors
\$FD30	Table of hardware and I/O vectors
\$FD50	Initialize work memory
\$FD98	Table of IRQ vectors
\$FDF9	Set parameter for file names
\$FE00	Set parameter for active file
\$FE07	Get status
\$FE1B	Set flag for messages of the operating system
\$FE1C	Set status
\$FE21	Set timeout flag for IEC-bus
\$FE25	Set or get RAM-upper limit
\$FE34	Set or get RAM-lower limit
\$FE43	NMI routine
\$FEC2	Constants for RS232 baud rate
\$FF4B	Interrupt handler

## JMP TABLE ADDRESSES

JMP ADDRESS	DESCRIPTION OF ROUTINE
\$FFB4	JMP \$FDA3 Initialize CIA's
\$FFB7	JMP \$FD50 Clear or check RAM
\$FFBA	JMP \$FD15 Initialize I/O
\$FFB0	JMP \$FD1A Initialize I/O vectors
\$FF90	JMP \$FE1B Set status
\$FF93	JMP \$E0B9 Send LISTEN secondary address
\$FF96	JMP \$E0C7 Send TALK Secondary address
\$FF99	JMP \$FE25 Set/get RAM end
\$FF9C	JMP \$FE34 Set/get RAM start
\$FF9F	JMP \$EAB7 Scan keyboard
\$FFA2	JMP \$FE21 Set IEC-bus time out flag
\$FFA5	JMP \$EE13 Input for IEC-bus
\$FFA8	JMP \$E0DD Output to IEC-bus
\$FFAB	JMP \$E0EF Send UNTALK
\$FFAE	JMP \$E0FE Send UNLISTEN
\$FFB1	JMP \$E0B9 Send LISTEN
\$FFB4	JMP \$E0C7 Send TALK
\$FFB7	JMP \$FE07 Get status
\$FFBA	JMP \$FE00 Set file parameter
\$FFBD	JMP \$FDF9 Set filename parameter
\$FFC0	JMP (\$031A) \$F34A OPEN
\$FFC3	JMP (\$031C) \$F291 CLOSE
\$FFC6	JMP (\$031E) \$F20E CHKIN set input device
\$FFC9	JMP (\$0320) \$F250 CKOUT set output device
\$FFCC	JMP (\$0322) \$F333 CLRC
\$FFCF	JMP (\$0324) \$F157 BASIN input character
\$FFD2	JMP (\$0326) \$F1CA BSOUT output character
\$FFD5	JMP \$F49E LOAD
\$FFD8	JMP \$F5D0 SAVE
\$FFDB	JMP \$F6E4 Set time
\$FFDE	JMP \$F6D0 Get time
\$FFE1	JMP (\$0328) \$F6ED Scan stop-key
\$FFE4	JMP (\$032A) \$F13E GET
\$FFE7	JMP (\$032C) \$F32F CLALL
\$FFEA	JMP \$F69B Increase time
\$FFED	JMP \$E505 SCREEN get number lines and columns
\$FFF0	JMP \$E50A Set/get cursor position
\$FFF3	JMP \$E500 Get start of I/O element
\$FFFA	JMP \$FE43 NMI vector
\$FFFC	JMP \$FCE2 RESET vector
\$FFFE	JMP \$FF4B IRQ vector

## 128 KERNEL ROUTINES

HEX ADDRESS	DESCRIPTION OF ROUTINE
\$C17C	Adapt attribute RAM address
\$B0FC	Addresses of the individual monitor cmds (table)
\$B8BA	Base table for four number systems
\$C98E	Bell: create tone
\$EFB4	BSOUT output not to screen
\$E224	C128 mode routine
\$FB1C	COMPARE routine for FAR operations RAM
\$FB1C	COMPARE routine for FAR operations ROM
\$EE9B	Change IRQ vector for tape operation
\$C3F4	Check Commodore key for time delay
\$F3A1	Check filename for burst mode
\$CA9F	Clear from cursor position to screen end
\$CA76	Clear from cursor position to line end
\$CA8B	Clear from line start to line end
\$CB81	Clear line overflow bit
\$C60A	Commodore/Shift character set switch
\$CB92	Commodore/Shift switch to 40-column mode
\$CB9F	Commodore/Shift switch to 80-column mode
\$E0CD	Copy NMI and IRQ routines to all banks
\$E723	CKOUT routine for RS-232 output
\$F16C	CKOUT evaluation on serial bus
\$F127	CHKIN evaluation on RS-232
\$E795	CHKIN routine for RS-232 input
\$F1A9	CLOSE routine for tape operation
\$EED0	Check cassette recorder-keyboard
\$E980	Check tape header address for validity
\$E242	Check EXROM input form cartridge test
\$E618	Check RS-232 send parity
\$CAEA	Clear or set auto-insert pointer
\$C142	Clear screen window
\$C4A5	Clear screen line in 40-column mode
\$C4C0	Clear screen line in 80-column mode
\$BBD2	Convert acc contents into two ASCII characters (X/A)
\$B8C2	Convert acc to two ASCII characters and output
\$F755	Coordinate system status word
\$E24B	Configure system as Commodore 64
\$C40D	Copy a window line (routine:MOVLIN)
\$C436	Copy a window line in 80-column mode
\$E051	Copy start address for input/output operations
\$F533	Control message: output LOADING
\$F50F	Control message: output SEARCHING FOR filename
\$F533	Control message: output VERIFYING
\$CE0C	Copy character set into VDC RAM
\$C320	Conversion from ASCII characters to POKE codes
\$C93D	Delete character under cursor
\$CA52	Delete current input line
\$CA24	Define screen as window
\$F1E4	Delete file entry from table
\$F1C1	Delete a file entry
\$C91B	Delete character to the left of the cursor
\$C3DC	Delete line on screen (with move)
\$B050	Display monitor register contents
\$B0C5	Determine address of a monitor command
\$B641	Determine address of BRANCH commands
\$C96C	Determine tab position
\$CBAB	Disable or enable Commodore/Shift
\$B3F0	DMA call routine of common area in RAM
\$CAFE	Enable block cursor
\$C194	Editor IRQ routine
\$C62F	Evaluate decoder table according to shift pattern
\$C6AD	Evaluate and store keypress
\$C7B6	Execute control code
\$C98E	Execute escape sequences
\$CBEB	Execute insert
\$B2A2	Fetch routine for FAR operations RAM
\$FB00	Fetch routine for FAR operations ROM
\$F7C9	Fetch routine for LSV operations
\$F7AE	Fetch routine for character from filename
\$C6E7	Flash VIC cursor
\$E26B	Function ROM test for C-128 mode
\$E569	Get bit from serial bus into carry flag
\$CC6A	Get cursor position and set
\$C244	Get character from keyboard queue
\$CB58	Get character and colour at cursor position
\$C29B	Get character from screen
\$EF5C	Get character from serial bus
\$EF4B	Get character from cassette
\$EF67	Get character from RS-232
\$E7CE	GET routine for RS-232
\$EEF9	GETIN evaluation not over keyboard
\$E5D6	Give fast-mode pulse on serial bus
\$E98E	Increment tape buffer pointer
\$C07B	Initialize screen and editor
\$C07B	Initialize editor and screen
\$B046	Initialization of monitor commands
\$B021	Initialize monitor for regular entry
\$B014	Initialize monitor after BREAK
\$E1DC	Initialize VDC registers
\$C37C	Insert line on screen
\$EAE8	Interrupt routine for tape read
\$ED90	Interrupt routine for tape write
\$CCF6	Insert function key string
\$B2E3	JMPFAR routine RAM
\$FB41	JMPFAR routine ROM
\$B2CD	JSRFAR routine RAM
\$FB2B	JSRFAR routine ROM
\$C94F	Jump to tab stop
\$E43E	Kernel Acptr routine
\$EF06	Kernel BASIN routine
\$F934	Kernel boot routine
\$EF79	Kernel BSOUT routine
\$F106	Kernel CHKIN routine
\$EF06	Kernel CHRIN routine
\$EF79	Kernel CHROUT routine
\$E503	Kernel CROUT routine
\$F14C	Kernel CKOUT routine
\$F222	Kernel CLALL routine
\$F18B	Kernel CLOSE routine
\$F226	Kernel CLRC routine
\$F7A5	Kernel DMA call routine
\$E5FB	Kernel FSTMODE routine
\$F7EC	Kernel GETCFG routine
\$EEEB	Kernel GETIN routine
\$E24B	Kernel G064 routine
\$F7B1	Kernel IOBASE routine
\$E109	Kernel IOINIT routine



\$FF17	Kernel IRQ routine
\$C55D	Kernel KEY routine (\$FCB7 in International versions)
\$E343	Kernel LISTN routine
\$F79D	Kernel LKUPLA routine
\$F7B6	Kernel LKUPSA routine
\$F265	Kernel LOAD routine
\$F772	Kernel MEMBOT routine
\$F763	Kernel MEMTOP routine
\$FF05	Kernel NMI routine
\$EFB0	Kernel OPEN routine
\$F867	Kernel PHOENIX routine
\$FA17	Kernel PRINM routine
\$E093	Kernel RANTAS routine
\$F65E	Kernel ROTIM routine
\$F744	Kernel READST routine
\$FF3D	Kernel RESET routine
\$E402	Kernel SECND routine
\$F73F	Kernel SETBNK routine
\$F73B	Kernel SETFLS routine
\$F75C	Kernel SETMSG routine
\$F731	Kernel SETNAM routine
\$F665	Kernel SETTIM routine
\$F75F	Kernel SETTMD routine
\$E388	Kernel TALK routine
\$E4E0	Kernel TKDA routine
\$F5FB	Kernel UDTIM routine
\$E526	Kernel UNLSN routine
\$E515	Kernel UNILK routine
\$E056	Kernel RESTOR routine
\$F53E	Kernel SAVE routine
\$F66E	Kernel STDP routine
\$E058	Kernel VECTOR routine
\$C67E	Key repeat evaluation
\$C55D	Keyboard matrix read
\$F63D	Keyboard row selection: RUN/STOP-SHIFT
\$C5E1	Keyboard read evaluate
\$C6CA	Keyboard buffer prepare for function key
\$B976	Load bank pointer and program counter from zero page
\$E9FB	Load program from cassette
\$F3EA	LOAD routine in burst mode
\$F27B	LOAD routine from serial bus
\$B406	Monitor command: (assemble a line)
\$B194	Monitor command: (change register)
\$B1AB	Monitor command: (change memory contents)
\$B9A0	Monitor command: (disk command)
\$B406	Monitor command: (assemble a line)
\$B231	Monitor command: (compare memory areas)
\$B599	Monitor command: (disassemble memory)
\$B30B	Monitor command: (fill memory area)
\$B106	Monitor command: (Jump to XXXX without return)
\$B2CE	Monitor command: (Search for memory contents)
\$B10F	Monitor command: (Jump to XXXX with RTS)
\$B337	Monitor command: (Load a program)
\$B152	Monitor command: (display memory contents)
\$B050	Monitor command: (display register contents)
\$B337	Monitor command: (store a program)
\$B234	Monitor command: (move memory areas)
\$B337	Monitor command: (compare program with memory)
\$B0E3	Monitor command: (exit)
\$B981	Monitor command: Convert number to different system
\$E805	NMI routine for RS-232
\$E8A9	NMI routine for RS-232 output
\$E878	NMI routine for RS-232 input
\$F915	Output boot sector message
\$F0C8	Open file on serial bus
\$EFF0	OPEN routine for tape operation
\$F040	OPEN routine for RS-232
\$E75C	Output in RS-232 buffer
\$CC2F	Output acc at cursor position
\$FD15	Output combined accent
\$CC27	Output space at cursor position
\$E3E2	Output byte on serial bus
\$C76F	Output carriage return to screen
\$C2BC	Output character at cursor position
\$C72D	Output character on screen
\$F521	Output found filename on screen
\$F71E	Output system and control messages
\$CEBC	Prepare byte output on serial bus
\$F9FB	Prepare acc contents in two ASCII characters (-99)
\$EA11	Prepare cassette synchronization
\$C363	Perform linefeed
\$E69D	Process received bit from RS-232
\$CCA2	Program function key
\$F4C5	Read data block in burst mode
\$E5F2	Read data block from tape
\$F4BA	Read data byte in burst mode
\$C258	Read an input line terminated by RETURN
\$E8D0	Read program header from cassette
\$E987	Recalculate tape-end address
\$EE57	Recorder operation end
\$EEB0	Recorder motor off
\$E080	Reset routine
\$C651	Repeat keyboard logic
\$C77D	Reset quote mode
\$F0B0	Reset CAs to RS-232
\$FCAA	Reset decoder table set vectors
\$F9B3	Recreate DOS output buffer
\$C9B0	Reset tab stops
\$E5FF	RS-232 output
\$E68E	RS-232 data-bit number calculate
\$E672	RS-232 NMI status set
\$E6D4	RS-232 start bittest
\$EFB7	RS-232 character output
\$C3A6	Scroll screen up
\$F5CB	SAVE routine for tape operation
\$E99A	Search tape header for name
\$C8C3	Search for end of input line
\$F282	Search in logical file number table
\$CACA	Scroll down
\$CABC	Scroll up
\$CAE2	Scrolling permit or prohibit
\$F23D	Set standard I/O devices
\$CA14	Set window borders
\$ED5A	Set bit counter for serial output
\$C837	Set or clear bell pointer
\$C0F9	Set attribute address for attribute RAM
\$C7E5	Set character colour in 40-column mode
\$C7EC	Set character colour in 80-column mode
\$C893	Set line overflow bit
\$C8D5	Set cursor flash mode
\$C057	Set cursor at current column

\$C33E	Set cursor to end of line
\$C158	Set cursor in screen window at HOME position
\$C875	Set cursor to left in window to left
\$C867	Set cursor up in window
\$C854	Set cursor right in window
\$C85A	Set cursor down in window
\$CC00	Set cursor one position left in window
\$C8ED	Set cursor one position right in window
\$C932	Set old cursor address again
\$C06F	Set cursor colour at cursor position
\$F8D5	Set filename to serial bus
\$C961	Set or clear tab stop
\$E573	Set clock frequency to 1MHz
\$C88F	Set or clear reverse mode
\$C207	Set IRQ register
\$F398	Set program end address after LOAD
\$02AF	Stash routine for FAR operations RAM
\$F8D0	Stash routine for FAR operations ROM
\$F7BC	Stash routine for LSV operations
\$C02C	Switch 40/80 column modes
\$FA65	System IRQ routine
\$FA40	System NMI routine
\$F7F0	Table of configuration values
\$CEB2	Table of function key assignments
\$C6D0	Table of function key codes
\$EEAB	Table of IRQ vectors for tape operations
\$CE74	Table of initialization values for 40-column
\$CE8E	Table of initialization values for 80-column
\$C78C	Table of control codes
\$E048	Table of MMU initialization values
\$B0E6	Table of monitor keywords
\$E850	Table of timer constants for RS-232 baud rate
\$E2FB	Table for VDC initialization
\$E2C7	Table for VIC initialization
\$FCC3	Test accent keys and combine accents
\$CB74	Test line overflow bit
\$C2FF	Test quote character and set pointer
\$B7A5	Test separator between command operands
\$EABF	Test for STOP key
\$E9DF	Test for tape button
\$C8DC	Turn off cursor flash mode
\$CB1A	Turn cursor flash off for 40 column mode
\$CB2E	Turn cursor flash on for 40 column mode
\$CB0B	Turn cursor flash off for 80 column mode
\$CB21	Turn cursor flash on for 80 column mode
\$CB48	Turn off 80 column reverse
\$CB3F	Turn on 80 column reverse
\$C8CE	Turn underline mode off
\$C8C7	Turn underline mode on
\$CAFE	Turn underline cursor on
\$C86F	Vector table to ASCII decoder tables
\$FE34	Vector table to DIN decoder tables (International versions only)
\$C000	Vector table for editor routines
\$C9DE	Vector table for escape routines
\$C786	Vector table for control code routines
\$F3EA	Verify routine in burst mode
\$EA7D	Wait for tape I/O termination
\$E7EC	Wait for end of RS-232 transfer
\$E58C	Wait for fast mode response from bus
\$E9E9	Wait for RECORD & PLAY on dataset
\$E9CA	Wait for button on dataset
\$EA15	Write tape buffer to tape
\$ED69	Write bit to tape
\$E919	Write data block to tape
\$E919	Write header to tape
\$EA1C	Write data block to tape
\$EE2E	Write the header

## USEFUL BASIC INTERPRETER ADDRESSES

HEX ADDRESS	DESCRIPTION OF ROUTINE
\$A000	Start vector
\$A002	NMI vector
\$A004	'CBMBASIC'
\$A00C	Addresses of the BASIC commands minus 1
\$A052	Addresses of the BASIC functions
\$A0B0	Hierarchy-codes and addresses of the BASIC operators
\$A09E	List of BASIC command words
\$A19E	BASIC error messages
\$A364	Messages of the BASIC interpreter
\$A38A	Stack search-routine for FOR-NEXT and GOSUB
\$A38B	Block-shifting routine
\$A3FB	Checks on space in stack
\$A40B	Makes space in memory
\$A435	Output of 'Out of memory'
\$A437	Output of error messages
\$A469	Break vector
\$A474	Ready vector
\$A480	Input waiting-loop
\$A49C	Clear and inserting program lines
\$A533	Tie BASIC program lines anew
\$A560	Gets a line into input buffer
\$A571	Output of 'String too long'
\$A579	Change of a line into interpreter-code
\$A613	Look for a start address of a BASIC line
\$A642	BASIC-command NEW
\$A65E	BASIC-command CLR
\$A68E	Set program pointer to BASIC start
\$A69C	BASIC-command LIST
\$A717	Change interpreter code to command word
\$A742	BASIC-command FOR
\$A7AE	Interpreter loop, carries out BASIC commands
\$A7ED	Carries out BASIC command
\$A81D	BASIC-command RESTORE
\$A82C	Interrupts program at pressed stop-key
\$A82F	BASIC-command STOP
\$A831	BASIC-command END
\$A857	BASIC-command CONT
\$A871	BASIC-command RUN
\$A883	BASIC-command GOSUB
\$A890	BASIC-command GOTO
\$A8FB	BASIC-command RETURN
\$A8D2	BASIC-command DATA
\$A906	Looks for next statement
\$A909	Looks for next line

```

$A92B BASIC-command IF
$A93B BASIC-command REM
$A94B BASIC-command ON
$A95B Looks for address of a BASIC line
$A9A5 BASIC-command LET
$A9B0 BASIC-command PRINT#
$A9B6 BASIC-command CMD
$A9A0 BASIC-command PRINT
$AB1E Output string
$AB3E Output empty character (Or cursor right)
$AB4D Error handling for INPUT
$AB7B BASIC-command GET
$ABAS BASIC-command INPUT#
$ABBF BASIC-command INPUT
$AC06 BASIC-command READ
$ACFC Output 'extra ignored' and 'redo from start'
$AD1D BASIC-command NEXT
$AD8A FRMNUM gets term and checks on numeric
$AD8D Checks on numeric
$AD8F Checks on string
$AD99 Output of 'Type mismatch'
$AD9E FRMNUM gets and evaluates term
$AE83 Get arithmetic term
$AEA0 Floating point constant for PI
$AED4 BASIC-command NOT
$AEF1 Gets term in parenthesis
$AEF7 Checks on Parenthesis closed
$AEFA Checks on Parenthesis open
$AEFD Checks on Commas
$AEFF Checks on characters in accumulator
$AF00 Output of 'Syntax error'
$AF2B Gets variable
$AF66 BASIC-command OR
$AF6E BASIC-command AND
$B016 Comparison operations
$B061 BASIC-command DIM
$B113 Checks for letter
$B194 Calculates pointer to first array element
$B1A5 Floating point constant -32768
$B1AA Change FAC to INTEGER
$B245 Output of 'Bad subscript'
$B246 Output of 'Illegal quantity'
$B34C Calculates array size
$B37D BASIC-function FRE
$B39E BASIC-function POS
$B3A6 Checks on direct-mode
$B3AB Output of 'Illegal direct'
$B3AE Output of 'Undef'd function'
$B3B3 BASIC-command DEF
$B3E1 Checks on FN syntax
$B3F4 BASIC-function FN
$B465 BASIC-function STR$
$B475 String administration, calculate pointer on string
$B4B7 Establish string
$B526 Garbage collection, remove unwanted strings
$B63D String connection '*'
$B6A3 String administration FRESTR
$B6EC BASIC-function CHR$
$B700 BASIC-function LEFT$
$B72C BASIC-function RIGHT$
$B737 BASIC-function MID$
$B77C BASIC-function LEN
$B782 Get string parameter
$B78B BASIC-function ASC
$B79B Gets byte term (0-255)
$B7AD BASIC-function VAL
$B7E8 Gets address (0-65535) and byte value (0-255)
$B7F7 Change FAC to address-format (Range 0-65535)
$B80D BASIC-function PEEK
$B824 BASIC-command POKE
$B82D BASIC-command WAIT
$B849 FAC = FAC + 0.5
$B850 Minus FAC = constant (A/Y) - FAC
$B853 Minus FAC = ARG - FAC
$B867 Plus FAC = constant (A/Y) + FAC
$B86A Plus FAC = ARG + FAC
$B87E Output of 'Overflow'
$B88C Floating point constant for LOG
$B89A BASIC-function LOG
$B8A9 Multiplication FAC = constant (A/Y) * FAC
$B8AB Multiplication FAC = ARG * FAC
$B8AC ARG = constant (A/Y)
$B8AE FAC = FAC * 10
$B8AF9 Floating point constant 10
$B8AF0 FAC = FAC/10
$B8BF FAC = constant (A/Y) / FAC
$B8B2 FAC = ARG/FAC
$B8BA Output of 'Division by zero'
$B8BA2 FAC = constant (A/Y)
$B8C4 Accu#4 = FAC
$B8CA Accu#3 = FAC
$B8D0 Variable = FAC
$B8DF FAC = ARG
$B8C0C ARG = FAC
$B8C1B Round FAC
$B8C2B Get sign of FAC
$B8C39 BASIC-function SGN
$B8C5B BASIC-function ABS
$B8C5B Compare constant (A/Y) with FAC
$B8C9B Change from FAC to integer
$B8CC BASIC-function INT
$B8CF3 Change ASCII to Floating point
$B8DB3 Floating point constants for Floating point to ASCII
$B8DC2 Output of line number at error message
$B8DCD Output of positive integer number (0-65535)
$B8DD0 Change FAC to ASCII format
$B8F11 Floating point constant 0.5
$B8F16 Binary numbers for change of FAC to ASCII
$B8F71 BASIC-function SQR
$B8F7B FAC = constant (A/Y) to the power of FAC
$B8F7B FAC = ARG to the power of FAC
$B8FBF Floating point constant for EXP
$B8FED BASIC-function EXP
$E043 Series 1 polynomial calculation
$E059 Series 2 polynomial calculation
$E08D Floating point constant for RND
$E097 BASIC-function RND
$E107 Output of 'Break'
$E10C BSOUT output of character
$E112 BASIN receive a character
$E11B CKOUT establish output-device

```

```

$E11E CKIN establish input-device
$E124 GETIN get a character
$E12A BASIC-command SYS
$E156 BASIC-command SAVE
$E165 BASIC-command VERIFY
$E16B BASIC-command LOAD
$E18E BASIC-command OPEN
$E1C7 BASIC-command CLOSE
$E1D4 Get parameters for LOAD and SAVE
$E219 Get parameter for OPEN
$E264 BASIC-function COS
$E26B BASIC-function SIN
$E2B4 BASIC-function TAN
$E2E0 Floating point constants for SIN and COS
$E30E BASIC-function ATN
$E33E Floating point constants for ATN
$E37B BASIC-MNI jump-in
$E394 BASIC cold start
$E3A2 Copy of the CHRGET routine
$E3BA Start value for the RND function
$E3BF Initialize RAM for BASIC
$E447 Table of BASIC vectors
$E453 Load BASIC vectors

```

## DISKETTE FORMATS & LAYOUTS

### BLOCK DISTRIBUTION BY TRACK

Track Numbers		Range of Sectors		Total sectors		S.Sided	D.Sided
HEX	DEC	HEX	DEC	HEX	DEC		
\$01-\$11	01-17	\$00-\$14	00-20	\$15	21	YES	YES
\$12-\$18	18-24	\$00-\$12	00-18	\$13	19	YES	YES
\$19-\$1E	25-30	\$00-\$11	00-17	\$12	18	YES	YES
\$1F-\$23	31-35	\$00-\$10	00-16	\$11	17	YES	YES
\$24-\$34	36-52	\$00-\$14	00-20	\$15	21	NO	YES
\$35-\$38	53-59	\$00-\$12	00-18	\$13	19	NO	YES
\$3C-\$41	60-65	\$00-\$11	00-17	\$12	18	NO	YES
\$42-\$46	66-70	\$00-\$10	00-16	\$11	17	NO	YES

### BAM FORMAT 1541 - TRACK 18 SECTOR 0

BYTE NUMBER	CONTENTS	DEFINITION
0	18	Track of next directory block. Always 18
1	1	Sector of next directory block. Always 1
2	65	ASCII character A indicating 1541/51/71/4040 format
3		DOS side flag. Ignored on 1541
4		Number of sector available on track 1
5		Track 1, sector 0-7 availability map
6		Track 1, sector 8-16 availability map
7		Track 1, sector 17-23 availability map
8		Number of sector available on track 2
9		Track 2, sector 0-7 availability map
10		Track 2, sector 8-16 availability map
11		Track 2, sector 17-23 availability map
...	...	...ETC DOWN TO...
140		Number of sector available on track 35
141		Track 35, sector 0-7 availability map
142		Track 35, sector 8-16 availability map
143		Track 35, sector 17-23 availability map
144-159		Disk name padded with shifted spaces [CHR\$(160)]
160-161	160	Shifted space [CHR\$(160)]
162-163		Disk ID
164	160	Shifted space [CHR\$(160)]
165-166		ASCII representation of 2A which are respectively the DOS version (2) format type 1540/41/51/71/4040/2030
167-170		Shifted spaces [CHR\$(160)]
171-255		Nulls [CHR\$(0)], not used
1571 Drive as above except:-		
3		Double sided flag: \$80=Double Sided, \$00=Single Sided
171-220		Nulls [CHR\$(0)], not used
221-237		Number of sector available track 36-52 (each sector by each byte)
238	0	Number of sector available track 53 (always 0, all sectors allocated)
239-244		Number sector available tk 54-59 (each tk by each byte)
245-250		Number sector available tk 60-65 (each tk by each byte)
251-255		Number sector available tk 66-70 (each tk by each byte)

### PROGRAM FILE FORMAT

BYTE	DEFINITION
FIRST SECTOR	
0,1	Track and sector of next block in program file 1
2,3	Load address of program
4-255	Next 252 bytes of prg info stored as in comp mem. (keywords tokenized)
REMAINING FULL SECTORS	
0,1	Track and sector of next block in program file 1
2-255	Next 254 bytes of prg info stored as in comp mem. (keywords tokenized)
FINAL SECTOR	
0,1	Null (\$00), followed by number of valid data bytes in sector
2-7??	Last bytes of prg info stored as in comp mem. (keywords tokenized). The end of a BASIC file is marked by three zero bytes in a row. Any remaining bytes in the sector are garbage and may be ignored.

### SEQUENTIAL FILE FORMAT

BYTE	DEFINITION
ALL BUT FINAL SECTOR	
0,1	Track and sector of next sequential data block
2-255	254 bytes of data



## FINAL SECTOR

0,1 Null (\$00), followed by number of valid data bytes in sector  
2-777 Last bytes of data. Any remaining bytes are garbage & can be ignored

## RELATIVE FILE FORMAT

### BYTE DEFINITION

#### DATA BLOCK

0,1 Track and sector of next data block  
2-255 254 bytes of data. Empty records contain \$FF (all binary ones) in the first byte followed by \$00 (all binary zeroes) to the end of the record. Partially filled records are padded with nulls (\$00)

#### SIDE SECTOR BLOCK

0-1 Track and sector of next side sector block  
2 Side sector number (0-5)  
3 Record length  
4-5 Track and sector of first side sector (number 0)  
6-7 Track and sector of second side sector (number 1)  
8-9 Track and sector of third side sector (number 2)  
10-11 Track and sector of fourth side sector (number 3)  
12-13 Track and sector of fifth side sector (number 4)  
14-15 Track and sector of sixth side sector (number 5)  
16-255 Track and sector pointers to 120 data blocks

## DIR FILE FORMAT, TRACK 18 SECTORS 1-19

### BYTE DEFINITION

0,1 Track and sector of next directory block  
2-31 File entry 1  
32-63 File entry 2  
64-95 File entry 3  
96-127 File entry 4  
128-159 File entry 5  
160-191 File entry 6  
192-223 File entry 7  
224-255 File entry 8

## STRUCTURE OF EACH INDIVIDUAL DIRECTORY ENTRY

### BYTE CONTENTS DEFINITION

0 128+type File type OR'ed with \$80 to indicate properly closed file. (if OR'ed with \$C0 instead, file is locked)  
TYPES: 0 = Deleted  
1 = Sequential  
2 = Program  
3 = User  
4 = Relative  
1-2 Track and sector of first data block  
3-18 File name padded with shifted spaces  
19-20 Rel file only. Track and sector of first side sector  
21 Rel file only. Record length  
22-25 UNUSED  
26-27 Track and sector of replacement file during an @SAVE or @OPEN  
28-29 Number of blocks in file, stored as a two-byte integer in normal 16-byte hi-byte format

## 1541 MEMORY MAP

### DRIVE ADDRESS

HEX	DECIMAL	DESCRIPTION
\$0000	0	Command code for buffer 0
\$0001	1	Command code for buffer 1
\$0002	2	Command code for buffer 2
\$0003	3	Command code for buffer 3
\$0004	4	Command code for buffer 4
\$0005-0007	5-7	Track and sector for buffer 0
\$0008-0009	8-9	Track and sector for buffer 1
\$000A-000B	10-11	Track and sector for buffer 2
\$000C-000D	12-13	Track and sector for buffer 3
\$000E-000F	14-15	Track and sector for buffer 4
\$0010-0011	16-17	ID for drive 0
\$0012-0013	18-19	ID for drive 1
\$0014-0015	20-21	ID
\$0016-0017	22-23	Flag for head transport
\$0018-0019	24-25	Buffer pointer for disk controller
\$001A-001B	26-27	Constant 0, mark for beginning of data block header
\$001C	28	Parity for data buffer
\$001D	29	Drive number for disk controller
\$001E	30	Buffer number for disk controller
\$001F	31	Number of sectors per track for formatting
\$0020	32	Constant 7, mark for beginning of data block header
\$0021	33	Stack pointer
\$0022	34	Step counter for head transport
\$0023	35	Actual track number for formatting
\$0024	36	Step size for sector division (10)
\$0025	37	Number of read attempts (5)
\$0026-0027	38-39	Pointer to address for M and B commands
\$0028	40	Device number + \$20 (32 dec) for Listen
\$0029	41	Device number + \$40 (64 dec) for Talk
\$002A	42	Flag for listen (I/O)
\$002B	43	Flag for talk (I/O)
\$002C	44	Flag for ATN from serial bus receiving
\$002D	45	Flag for EOI from serial bus
\$002E	46	Drive number
\$002F	47	Track number
\$0030	48	Sector number
\$0031	49	Channel number
\$0032	50	Secondary address
\$0033	51	Secondary address
\$0034	52	Data byte
\$0035-0036	53-54	Work storage for division
\$0037-0038	55-56	Actual buffer pointer
\$0039-003A	57-58	Address of buffer 0 \$0300
\$003B-003C	59-60	Address of buffer 1 \$0400
\$003D-003E	61-62	Address of buffer 2 \$0500
\$003F-0040	63-64	Address of buffer 3 \$0600
\$0041-0042	65-66	Address of buffer 4 \$0700
\$0043-0044	67-68	Pointer to input buffer \$0200
\$0045-0046	69-70	Pointer to buffer error message \$0205
\$0047-0048	71-72	Record number L0, block number L0
\$0049-004A	73-74	Record number M1, block number M1
\$004B-004C	75-76	Write pointer for REL file
\$004D-004E	77-78	Record length for REL file

\$0004	212	Pointer in record for REL file
\$0005	213	Side sector number
\$0006	214	Pointer to data block in side sector
\$0007	215	Pointer to record in REL file
\$0008	231	File type
\$0009	249	Buffer number
\$0100-0145	256-325	Stack
\$0200-0220	512-552	Buffer for command string
\$024A	586	File type
\$0258	600	Record length
\$0259	601	Track side-sector
\$025A	602	Sector side-sector
\$0274	628	Length of input line
\$0278	632	Number of file names
\$0297	663	File control method
\$02B0-02B4	640-644	Track of a file
\$02B5-02B9	645-649	Sector of a file
\$02D5-02F9	725-761	Buffer for error messages
\$02FA-02FC	762-764	Number of free blocks
\$0300-03FF	768-1023	Buffer 0
\$0400-04FF	1024-1279	Buffer 1
\$0500-05FF	1280-1535	Buffer 2
\$0600-06FF	1536-1791	Buffer 3
\$0700-07FF	1792-2047	Buffer 4

## 1541 DISK ERROR MESSAGES AND THEIR CAUSES

The following list are the error messages that the DOS can recognise. Note that TT and SS denote Track and Sector respectively.

ERROR NUMBER	DESCRIPTION
00,0X,00,00	This message occurs when the last disk operation was error free or if no command or data was sent after the last error message.
20,READ ERROR,TT,SS	This error means that the 'header' of a block was not found. It is usually the result of a defective disk. TT and SS designate the track and sector in which the error occurred. Remedy: change the disk.
21,READ ERROR,TT,SS	The SYNC marker of a block was not found. The cause may be an unformatted disk, or no disk in the drive. This error can also be caused by misaligned read/write head. Remedy: Either insert a diskette and format the disk, or have the heads aligned.
22,READ ERROR,TT,SS	This error means that a checksum error has occurred in the header of a data block, which can be caused by the incorrect writing of a block.
23,READ ERROR,TT,SS	This error implies that a data block was read into the DOS buffer, but a checksum error occurred. One or more data bytes are incorrect. Remedy: Save as many files as possible onto another diskette.
24,READ ERROR,TT,SS	This error also results from a checksum error in the data block or in the preceding data header. Incorrect bytes have been read. Remedy: Same as 23
25,WRITE ERROR,TT,SS	This is actually a VERIFY error. After writing every block the data is read again, checked against the data in the buffer. This error is produced if the data are not identical. Remedy: Repeat the command that caused the error. If this does not work, the corresponding block must be locked out from further use by the Block-allocate command.
26,WRITE PROTECT ON,TT,SS	An attempt was made to write to a disk with the write protect tab on. Remedy: Remove the tab.
27,READ ERROR,TT,SS	A checksum error occurred in the header of a data block. Remedy: Repeat command or rescue block.
28,WRITE ERROR,TT,SS	After writing a data block, the SYNC characters of the next data block were not found. Remedy: Format the disk again, or exchange it.
29,DISK IO MISMATCH,TT,SS	The ID in the DOS memory does not agree with the ID on the diskette. The diskette was either not initialized or there is an error in the header of a data block. Remedy: Initialize diskette.
30,SYNTAX ERROR,00,00	A command was sent over the command channel that the DOS could not understand. Remedy: Correct cmd.
31,SYNTAX ERROR,00,00	A command was not recognized by the DOS. Remedy: Do not use the command.
32,SYNTAX ERROR,00,00	The command sent over the command channel was over 40 characters. Remedy: Shorten the command.
33,SYNTAX ERROR,00,00	A wildcard, ("*" or "?") was used in an OPEN or SAVE command. Remedy: Remove wildcard.
34,SYNTAX ERROR,00,00	The DOS cannot find the filename in a command. The cause maybe a colon was forgotten after the cmd. word. Remedy: Check command.
39,FILE NOT FOUND,00,00	User program of type "USR" was not found for automatic execution. Remedy: Check filename.
50,RECORD NOT PRESENT,00,00	A record was addressed in a relative data file that has not yet been written. When writing a record this is not really an error. You can avoid this error message if you write the highest record number of the file with CHR\$(255) when initializing it. This error will no longer occur upon later access.
51,OVERFLOW IN RECORD,00,00	The number of characters sent when writing a record in a relative file was greater than the record length. The excess characters are ignored.
52,FILE TOO LARGE,00,00	The record number of a relative file is too big; the diskette does not have enough capacity. Remedy: Use another diskette or reduce the record number.

60,WRITE FILE OPEN,00,00	An attempt was made to OPEN a file that had not been previously been CLOSED after writing. Remedy: Use mode 'M' in the OPEN command to read file.
61,FILE NOT OPEN,00,00	A file was accessed that had not been OPENed. Remedy: OPEN the file or check the filename.
62,FILE NOT FOUND,00,00	An attempt was made to load a program or open a file that does not exist on the disk. Remedy: Check the filename.
63,FILE EXISTS,00,00	An attempt was made to establish a new file with the name of an existing file on the disk. Remedy: Use different name or use 00.
64,FILE TYPE MISMATCH,00,00	The file type used in the OPEN command does not agree with the file type in the directory. Remedy: Correct filetype.
65,NO BLOCK,IT,SS	This message is given in association with the Block-allocate command when the specified block is no longer free. In this case, the DOS automatically searches for a free block with a higher sector and/or track number and gives these values as the track and sector number in the error message. If no block with a greater number is free, two zeroes will be given.
66,ILLEGAL II or SS,IT,SS	If you attempt to use a block with the block commands that does not exist, this error is then returned.
67,ILLEGAL II or SS,IT,SS	The track-sector combination of a file produces a non-existent track or sector.
70,NO CHANNEL,00,00	An attempt was made to open more files than channels available or a direct access channel is already reserved.
71,DIR ERROR,IT,SS	The number of free blocks in the DOS storage does not agree with the BAM. Usually this means the disk has not been initialized.
72,DISK FULL,00,00	Fewer than three blocks are free on the disk or the maximum number of directory entries have been used. (144 on the 1541).
73,CBM DOS v.26 1541,00,00	The message is the power-up message of the 1541. As an error message, it appears when an attempt is made to write to a disk that was not formatted with the same DOS version.
74,DRIVE NOT READY,00,00	The drive does not have a diskette inserted.
75,FORMAT SPEED ERROR,00,00	This error only occurs on the CBM 8250. It indicates a deviation from the normal revolution speed whilst formatting.

## STANDARD CBM KEYWORD TOKENS

HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
\$20	32	SPACE	\$43	67	C	\$86	134
\$21	33	"	\$44	68	D	\$87	135
\$22	34	"	\$45	69	E	\$88	136
\$23	35	#	\$46	70	F	\$89	137
\$24	36	\$	\$47	71	G	\$8A	138
\$25	37	%	\$48	72	H	\$8B	139
\$26	38	&	\$49	73	I	\$8C	140
\$27	39	'	\$4A	74	J	\$8D	141
\$28	40	(	\$4B	75	K	\$8E	142
\$29	41	)	\$4C	76	L	\$8F	143
\$2A	42	*	\$4D	77	M	\$90	144
\$2B	43	+	\$4E	78	N	\$91	145
\$2C	44	,	\$4F	79	O	\$92	146
\$2D	45	-	\$50	80	P	\$93	147
\$2E	46	.	\$51	81	Q	\$94	148
\$2F	47	/	\$52	82	R	\$95	149
\$30	48	0	\$53	83	S	\$96	150
\$31	49	1	\$54	84	T	\$97	151
\$32	50	2	\$55	85	U	\$98	152
\$33	51	3	\$56	86	V	\$99	153
\$34	52	4	\$57	87	W	\$9A	154
\$35	53	5	\$58	88	X	\$9B	155
\$36	54	6	\$59	89	Y	\$9C	156
\$37	55	7	\$5A	90	Z	\$9D	157
\$38	56	8	\$5B	91	[	\$9E	158
\$39	57	9	\$5C	92	]	\$9F	159
\$3A	58	:	\$5D	93	{	\$A0	160
\$3B	59	;	\$5E	94	}	\$A1	161
\$3C	60	<	\$5F	95	~	\$A2	162
\$3D	61	=	\$60	128	END	\$A3	163
\$3E	62	>	\$61	129	FOR	\$A4	164
\$3F	63	?	\$62	130	NEXT	\$A5	165
\$40	64	@	\$63	131	DATA	\$A6	166
\$41	65	A	\$64	132	INPUT#	\$A7	167
\$42	66	B	\$65	133	INPUT	\$A8	168

## STANDARD CBM128 TOKENS

HEX	DEC	HEX	DEC	HEX	DEC
\$80	128	END	\$AA	170	+
\$81	129	FOR	\$AB	171	-
\$82	130	NEXT	\$AC	172	*
\$83	131	DATA	\$AD	173	/
\$84	132	INPUT#	\$AE	174	(POWER)
\$85	133	INPUT	\$AF	175	AND
\$86	134	DIM	\$B0	176	OR
\$87	135	READ	\$B1	177	>
\$88	136	LET	\$B2	178	=
\$89	137	GOTO	\$B3	179	<
\$8A	138	RUN	\$B4	180	SGN
\$8B	139	IF	\$B5	181	INT
\$8C	140	RESTORE	\$B6	182	ABS
\$8D	141	GOSUB	\$B7	183	USR

\$8E	142	RETURN	\$8B	184	FRE	\$E3	227	GSHAPE
\$8F	143	REM	\$89	185	POS	\$E4	228	SSHAPE
\$90	144	STOP	\$8A	186	SQR	\$E5	229	DRAW
\$91	145	ON	\$8B	187	RND	\$E6	230	LOCATE
\$92	146	WAIT	\$8C	188	LOG	\$E7	231	COLOR
\$93	147	LOAD	\$8D	189	EXP	\$E8	232	SCNCLR
\$94	148	SAVE	\$8E	190	COS	\$E9	233	SCALE
\$95	149	VERIFY	\$8F	191	SIN	\$EA	234	HELP
\$96	150	DEF	\$C0	192	TAN	\$EB	235	DO
\$97	151	POKE	\$C1	193	ATN	\$EC	236	LOOP
\$98	152	PRINT#	\$C2	194	PEEK	\$ED	237	EXIT
\$99	153	PRINT	\$C3	195	LEN	\$EE	238	DIRECTORY
\$9A	154	CONT	\$C4	196	STR\$	\$EF	239	DSAVE
\$9B	155	LIST	\$C5	197	VAL	\$F0	240	DLOAD
\$9C	156	CHD	\$C6	198	ASC	\$F1	241	HEADER
\$9D	157	CLR	\$C7	199	CHR\$	\$F2	242	SCRATCH
\$9E	158	SYS	\$C8	200	LEFT\$	\$F3	243	COLLECT
\$9F	159	OPEN	\$C9	201	RIGHT\$	\$F4	244	COPY
\$A0	160	CLOSE	\$CA	202	MID\$	\$F5	245	RENAME
\$A1	161	GET	\$CB	203	GO	\$F6	246	BACKUP
\$A2	162	NEW	\$CC	204	RGR	\$F7	247	DELETE
\$A3	163	TAB(	\$CD	205	RCLR	\$F8	248	RENUMBER
\$A4	164	TO	\$CE	206	JOY	\$F9	249	KEY
\$A5	165	FN	\$D0	208	ROOT	\$FA	250	MONITOR
\$A6	166	SPC(	\$D1	209	DEC	\$FB	251	USING
\$A7	167	THEN	\$D2	210	HEX\$	\$FC	252	UNTIL
\$A8	168	NOT	\$D3	211	ERR\$	\$FD	253	WHILE
\$A9	169	STEP	\$D4	212	INSTR			

## 128 DOUBLE-BYTE TOKENS

HEX	DEC	HEX	DEC
\$CE	\$02	206	2
\$CE	\$03	206	3
\$CE	\$04	206	4
\$CE	\$05	206	5
\$CE	\$06	206	6
\$CE	\$07	206	7
\$CE	\$08	206	8
\$CE	\$09	206	9
\$CE	\$0A	206	10
\$FE	\$02	254	2
\$FE	\$03	254	3
\$FE	\$04	254	4
\$FE	\$05	254	5
\$FE	\$06	254	6
\$FE	\$07	254	7
\$FE	\$08	254	8
\$FE	\$09	254	9
\$FE	\$0A	254	10
\$FE	\$0B	254	11
\$FE	\$0C	254	12
\$FE	\$0D	254	13
\$FE	\$0E	254	14
\$FE	\$0F	254	15
\$FE	\$10	254	16

## PLUS4 ABBREVIATED KEYWORDS

KEYWORD	ABBREVIATION	KEYWORD	ABBREVIATION	KEYWORD	ABBREVIATION
ABS	AshifTB	GOSUB	GoshifTS	READ	RshifTE
ASC	AshifTS	GOTO	GshifTD	RENAME	REshifTN
ATN	AshifTI	GRAPHIC	GshifTR	RENUMBER	REshifTU
AUTO	AshifTU	GSHAPE	GshifTS	RENUMBER	REshifTU
BACKUP	BshifTA	HEADER	HshifTA	RESTORE	REshifTS
BOX	BshifTD	HEX\$	HshifTE	RESUME	REshifTU
CHAR	ChshifTA	INPUT#	IshifTN	RETURN	REshifTI
CHR\$	ChshifT	INSTR	InshifTD	RGR	RshifTG
CIRCLE	ChshifTI	JOY	JshifTD	RIGHT\$	RshifTI
CLOSE	ClshifTD	KEY	KshifTE	RUM	RshifTL
CLR	ChshifTL	LEFT\$	LEshifTF	RND	RshifTN
CMD	ChshifTN	LET	LshifTE	RUN	RshifTU
COLLECT	COLshifTL	LIST	LshifTI	SAVE	SshifTA
COLOR	COshifTL	LOAD	LshifTD	SCALE	SCshifTA
CONT	ChshifTD	LOCATE	LOshifTC	SCNCLR	SshifTC
COPY	COshifTP	LOOP	LOshifTD	SCRATCH	SCshifTR
DATA	DshifTA	MID\$	MshifTI	SGN	SshifTG
DEF FN	DshifTE	MONITOR	MshifTD	SIN	SshifTI
DELETE	DEshifTL	NEXT	NshifTE	SOUND	SshifTD
DIM	DshifTI	ON...GOSUB	ON...GoshifTS	SPC(	SshifTP
DIRECTORY	DshifTR	ON...GOTO	ON...GoshifTD	SQR	SshifTG
DLOAD	DshifTL	OPEN	OshifTP	SSHAPE	SshifTS
DRAW	DshifTR	PAINT	PshifTA	STOP	SshifTI
DSAVE	DshifTS	PEEK	PshifTE	STR\$	StshifTR
END	EshifTN	POKE	PshifTD	SYS	SshifTV
ERR\$	EshifTR	PRINT	?shifT	TAB(	TshifTA
EXP	EshifTX	PRINT#	PshifTR	TRAP	TshifTR
FOR	FshifTD	PRINT USING	?USshifTI	TROFF	TRshifTS
FRE	FshifTR	PUDEF	PshifTU	TRON	TRshifTD
GET	GshifTE	RCLR	RshifTC	UNTIL	UshifTN
GETKEY	GETKshifTE	ROOT	RshifTD	USR	UshifTS

NOTE:- Any keywords not listed indicates that there is not abbreviation for that particular keyword.

## 128 ABBREVIATED KEYWORDS

KEYWORD	ABBREVIATION	KEYWORD	ABBREVIATION	KEYWORD	ABBREVIATION
ABS	AshifTB	FRE	FshifTR	RENUMBER	REshifTU
APPEND	AshifTP	GET	GshifTE	RESTORE	REshifTS
ASC	AshifTS	GETKEY	GETKshifTE	RESUME	REshifTU
ATN	AshifTI	GOSUB	GoshifTS	RETURN	REshifTI
AUTO	AshifTU	GOTO	GshifTD	RGR	RshifTG
BACKUP	BshifTA	GRAPHIC	GshifTR	RIGHT\$	RshifTI
BANK	BshifTS	GSHAPE	GshifTS	RND	RshifTN
BEGIN	BshifTE	HEADER	HshifTA	RREG	RshifTR
BEND	BshifTN	HELP	HshifTL	RSPCOLOR	RSPshifTC
BLOAD	BshifTL	HEX\$	HshifTE	RSPPOS	RshifTS
BOOT	BshifTD	INPUT#	IshifTN	RSPRITE	RSPshifTR



BSAVE	BshiftS	INSTR	InshiftS	RUN	RshiftU
BUMP	BshiftU	JOY	JshiftO	RWINDOW	RshiftW
CATALOG	CshiftA	KEY	KshiftE	SAVE	SshiftA
CHAR	ChshiftA	LEFTS	LshiftF	SCALE	SCshiftA
CHRS	CshiftX	LET	LshiftE	SCNCLR	SCshiftC
CIRCLE	CshiftI	LIST	LshiftI	SCRATCH	SCshiftR
CLOSE	ClshiftO	LOAD	LshiftO	SGN	SshiftG
CLR	CshiftL	LOCATE	LoShiftC	SIN	SshiftI
CMD	CshiftM	LOOP	LoShiftO	SLEEP	SshiftL
COLLECT	COLshiftE	MIDS	MshiftI	SOUND	SshiftO
COLLISION	COLshiftL	MONITOR	MOshiftN	SPRCOLOR	SPRshiftC
COLOR	COLshiftO	MOVSPR	MshiftO	SPRDEF	SPRshiftD
CONCAT	CshiftO	NEXT	NshiftE	SPRITE	SshiftP
COPY	COshiftP	ON..GOSUB	ON..GOshiftS	SPRSAY	SPRshiftS
DATA	DshiftA	ON..GOTO	ON..GOshiftO	SQR	SshiftQ
DCLEAR	DCLshiftE	OPEN	OshiftP	SSHAPE	SshiftS
DCLOSE	DshiftC	PAINI	PshiftA	STASH	SshiftT
DELETE	DEshiftL	PEEK	PshiftE	STEP	STshiftE
DIN	DshiftI	PEN	PshiftE	STOP	STshiftO
DIRECTORY	DshiftR	PLAY	PshiftL	STR\$	STshiftR
DLOAD	DshiftL	POKE	POshiftO	SWAP	SshiftW
DOPEN	DshiftO	POT	PshiftO	TAB	TshiftA
DRAW	DshiftR	PRINT	PshiftR	TEMP	TshiftE
DSAVE	DshiftS	PRINT#	PshiftR	TRAP	TshiftR
DVERIFY	DshiftV	PRINT USING	?USshiftI	TROFF	TROshiftF
ENVELOPE	EshiftN	PUDEF	PshiftU	TRON	TRshiftO
ERR\$	EshiftR	RBUMP	RBshiftU	UNTIL	UshiftN
EXIT	EXshiftI	RCLR	RshiftC	USR	UshiftS
EXP	EshiftX	RDOT	RshiftO	VERIFY	VshiftE
FETCH	FshiftE	READ	REshiftA	VOL	VshiftO
FILTER	FshiftI	RECORD	REshiftE	WAIT	WshiftA
FOR	FshiftO	RENAME	REshiftN	WHILE	WshiftH
				WIDTH	WshiftI
				WINDOW	WshiftI
				XOR	XshiftO

NOTE:- Any keywords not listed indicates that there is not abbreviation for that particular keyword.

## PLUS4 COLOUR CODES

Value to POKE for each colour:

COLOUR	VALUE
Black	1
White	2
Red	3
Cyan	4
Purple	5
Green	6
Blue	7
Yellow	8
Orange	9
Brown	10
Yellow-green	11
Pink	12
Blue-green	13
Light blue	14
Dark blue	15
Light green	16

## C64 SCN/COLOUR CODES & MODES

Value to POKE for each colour:

COLOUR	LOW NYBBLE COLOUR VALUE	HIGH NYBBLE COLOUR VALUE	MULTICOLOUR COLOUR VALUE
Black	0	0	8
White	1	16	9
Red	2	32	10
Cyan	3	48	11
Purple	4	64	12
Green	5	80	13
Blue	6	96	14
Yellow	7	112	15
Orange	8	128	---
Brown	9	144	---
Lt Red	10	160	---
Dark Gray	11	176	---
Med Gray	12	192	---
Lt Green	13	208	---
Lt Blue	14	224	---
Lt Gray	15	240	---

Where to POKE colour values for each mode:

MODE [0]	BIT OR BIT-PAIR	LOCATION	COLOUR VALUE
Regular text	0	53281	Low nybble
	1	Colour memory	Low nybble
Multicolour text	00	53281	Low nybble
	01	53282	Low nybble
	10	53283	Low nybble
	11	Colour memory	Select multicolour
Extended colour text	00	53281	Low nybble
	01	53282	Low nybble
	10	53283	Low nybble
	11	53284	Low nybble
Bitmapped	0	Screen memory	Low nybble [2]
	1	Screen memory	High nybble [2]
Multicolour bitmapped	00	53281	Low nybble [2]
	01	Screen memory	High nybble [2]
	10	Screen memory	Low nybble [2]
	11	Colour memory	Low nybble

- [0] For all modes, the screen border colour is controlled by POKE'g 53280 with the low nybble colour value.
- [1] In extended colour mode, bits 6 & 7 of each byte of screen memory serve as the bit-pair controlling background colour. Because only bits 0-5 are available for character selection, only characters with screen codes 0-63 can be used in this mode.
- [2] In the bitmapped modes, the high and low nybble colour values are ORed together and POKE'd into the SAME LOCATION in screen memory to control the colours of the corresponding CELL in the bitmap. For example: to control the colours of cell 0 of the bitmap, OR the high and low nybble values and POKE the result into location 0 of screen memory.

## C128 COLOUR CODES

### COLOUR SOURCE

NUMBER	SOURCE
0	40 column background colour (VIC)
1	Foreground for graphics screen (VIC)
2	Foreground colour 1 for multicolour screen (VIC)
3	Foreground colour 2 for multicolour screen (VIC)
4	40 column border (VIC - whether in text or graphics mode)
5	Character colour for 40 or 80 column text screen
6	80 column background colour (8563)

Value to POKE for each colour: 40 column mode:

COLOUR	VALUE
Black	1
White	2
Red	3
Cyan	4
Purple	5
Green	6
Blue	7
Yellow	8
Orange	9
Brown	10
Light red	11
Dark gray	12
Medium gray	13
Light green	14
Light blue	15
Light gray	16

Value to POKE for each colour: 80 column mode:

COLOUR	VALUE
Black	1
White	2
Dark red	3
Light cyan	4
Light purple	5
Dark green	6
Dark blue	7
Light yellow	8
Dark purple	9
Brown	10
Light red	11
Dark cyan	12
Medium gray	13
Light green	14
Light blue	15
Light gray	16

## CBM CONTROL & CHR\$ CODES

CHR\$	KEYS	FUNCTION	C64	C128
CHR\$(2)	CTRL B	Underline (80 col)	N	Y
CHR\$(5)	CTRL 2/CTRL E	Set char color to white (80 and 40 col)	Y	Y
CHR\$(7)	CTRL G	Produce bell tone	N	Y
CHR\$(8)	CTRL H	Disable char set change key	Y	N
CHR\$(9)	CTRL I	Enable char set change key	Y	N
		Move cursor to next tab position	N	Y
CHR\$(10)	CTRL J	Send carriage return with line feed	Y	N
		Send a line feed	N	Y
CHR\$(11)	CTRL K	Disable char set change key	N	Y
CHR\$(12)	CTRL L	Enable char set change key	N	Y
CHR\$(13)	CTRL M	Send carr ret, line feed & enter BASIC line	Y	Y
CHR\$(14)	CTRL N	Set char set to lower/upper case set	Y	Y
CHR\$(15)	CTRL O	Turn flash on (80 col)	N	Y
CHR\$(17)	CTRL Q/CRSRD	Move cursor down one row	Y	Y
CHR\$(18)	CTRL B	Chars to be printed in reverse field	Y	Y
CHR\$(19)	CTRL S/HOME	Move cursor to home position	Y	Y
CHR\$(20)	CTRL I/DEL	Delete last char typed	Y	Y
CHR\$(27)	CTRL /ESC	Send an ESC character	N	Y
CHR\$(28)	CTRL 3/CTRL /	Make char color RED (40 and 80 column)	Y	Y
CHR\$(29)	CTRL 3/CRSR	Move cursor one column to right	Y	Y
CHR\$(30)	CTRL 6	Set char color to GREEN (40 and 80 column)	Y	Y
CHR\$(34)	"	Print double quote on scn & set quote mode	Y	Y
CHR\$(129)	CBM 1	Set char color ORANGE (40 col), DARK PURPLE (80 col)	Y	Y
CHR\$(130)		Underline off (80 column)	N	Y
CHR\$(131)		Run a programme. (Only works from kb buffer)	Y	Y
CHR\$(133)	F1	Reserved CHR\$ code for F1 key	Y	Y
CHR\$(134)	F3	Reserved CHR\$ code for F3 key	Y	Y
CHR\$(135)	F5	Reserved CHR\$ code for F5 key	Y	Y
CHR\$(136)	F7	Reserved CHR\$ code for F7 key	Y	Y
CHR\$(137)	F2	Reserved CHR\$ code for F2 key	Y	Y
CHR\$(138)	F4	Reserved CHR\$ code for F4 key	Y	Y
CHR\$(139)	F6	Reserved CHR\$ code for F6 key	Y	Y
CHR\$(140)	F8	Reserved CHR\$ code for F8 key	Y	Y

CHRS(141) SHIFT RETURN	Send carriage return and line feed without entering a line of basic	Y	Y
CHRS(142)	Set char set to upper case set	Y	Y
CHRS(143)	Turn flash off (80 column)	N	Y
CHRS(144) CTRL 1	Set char color to BLACK (40 and 80 col)	Y	Y
CHRS(145) CRSR UP	Move cursor or printing up one row	Y	Y
CHRS(146) CTRL 0	Terminate reverse field mode	Y	Y
CHRS(147) HOME	Clear window screen & position csr topleft	Y	Y
CHRS(148) INST	Move char from cursor position right 1 col	Y	Y
CHRS(149) CBM 2	Set char color BROWN (40) DARK YELLOW (80)	Y	Y
CHRS(150) CBM 3	Set char color LIGHT RED (40 and 80 column)	Y	Y
CHRS(151) CBM 4	Set char color DARK GRAY (40) DARKCYAN (80)	Y	Y
CHRS(152) CBM 5	Set char color MEDIUM GRAY (40 and 80)	Y	Y
CHRS(153) CBM 6	Set char color LIGHT GREEN (40 and 80)	Y	Y
CHRS(154) CBM 7	Set char color LIGHT BLUE (40 and 80)	Y	Y
CHRS(155) CBM 8	Set char color LIGHT GRAY (40 and 80)	Y	Y
CHRS(156) CTRL 5	Set char color PURPLE (40 and 80)	Y	Y
CHRS(157) CRSR LEFT	Move cursor left one column	Y	Y
CHRS(158) CTRL 4	Set char color CYAN (40 and 80)	Y	Y

## 128/PLUS4 ESCAPE CODES

ESCAPE KEYS	ESCAPE FUNCTION	PLUS4	128
ESC A	Enable Auto insert mode	Y	Y
ESC B	Set bottom of screen window at cursor position	Y	Y
ESC C	Disable auto insert mode	Y	Y
ESC D	Delete current line	Y	Y
ESC E	Set cursor to non flashing mode	N	Y
ESC F	Set cursor to flashing mode	N	Y
ESC G	Enable bell (by control-G)	N	Y
ESC H	Disable bell	N	Y
ESC I	Insert a line	Y	Y
ESC J	Move to start of current line	Y	Y
ESC K	Move to end of current line	Y	Y
ESC L	Turn on scrolling	Y	Y
ESC M	Turn off scrolling	Y	Y
ESC N	Return to normal screen display size (128 in 80 col)	Y	Y
ESC O	Cancel insert, quote, reverse and flash modes	Y	Y
ESC P	Erase all up to start of current line	Y	Y
ESC Q	Erase all to end of current line	Y	Y
ESC R	Reduce screen display	Y	N
ESC R	Set screen to reverse video (128 80 col only)	N	Y
ESC S	Change to block cursor (128 80 col)	N	Y
ESC T	Set top of screen window to top left corner	Y	Y
ESC U	Change to underline cursor (128 80 col only)	N	Y
ESC U	Scroll screen up	Y	Y
ESC W	Scroll screen down	Y	Y
ESC X	Cancel esc function (plus 4 only)	Y	N
ESC X	Swap 40/80 column display output device	N	Y
ESC Y	Set default tab stop (8 spaces)	N	Y
ESC Z	Clear all tab stops	N	Y
ESC 0	Clear to end of screen	N	Y

## CB4 KEYCODES STORED AT LOCATION 197

KEY	KEYCODE	KEY	KEYCODE
A	10	6	19
B	20	7	24
C	20	8	27
D	18	9	32
E	14	0	35
F	21	+	40
G	26	-	43
H	29	=	48
I	33	CLR/HOME	51
J	34	INST/DEL	0
K	37	Lt.ARROW	57
L	42	*	46
M	36	.	49
N	39	:	54
O	38	;	45
P	41	'	50
Q	62	~	53
R	17	RET	1
S	13	,	47
T	22	/	44
U	30	/	55
U	31	CSR UP/DOWN	7
W	9	CSR LT/RT	2
X	23	F1	4
Y	25	F3	5
Z	12	F5	6
1	56	F7	3
2	59	SPACE	60
3	8	RUNSTOP	63
4	11	NOKEY PRESSED	64
5	15		

Values to be found at location 653

CODE	KEY(s) PRESSED
0	No key pressed
1	SHIFT
2	Commodore
3	SHIFT and Commodore
4	CTRL
5	Shift and CTRL
6	Commodore and CTRL
7	SHIFT, CTRL and Commodore

## +4 TED CHIP - MEMORY MAP

REG	HEX ADD	DEC ADD	BITS	DESCRIPTION
0	\$FF00	65280		Timer 1, Reload low
1	\$FF01	65281		Timer 1, Reload high
2	\$FF02	65282		Timer 2, low
3	\$FF03	65283		Timer 2, high
4	\$FF04	65284		Timer 3, low
5	\$FF05	65285		Timer 3, high
6	\$FF06	65286	0-2	Vertical scroll position (Y)
			3	Select 24/25 rows (1-25)
			4	Switch screen off
			5	Switch bitmap mode (1=ON)
			6	Switch extended color mode (1=ON)
			7	Testbit (always 0)
7	\$FF07	65287	0-2	Horizontal scroll position (X)
			3	Select 38/40 columns (1=40)
			4	Switch multicolor mode (1=ON)
			5	Switch freeze mode on (1=ON)
			6	PAL/NTSC mode (0=PAL, 1=NTSC)
			7	RUS video (0=hardware, 1=software)
8	\$FF08	65288		Keyboard matrix
9	\$FF09	65289		Interrupt sources
			0	Not used
			1	Raster interrupt
			2	Light pen (Not possible with C-16)
			3	Timer 1 interrupt
			4	Timer 2 interrupt
			5	Not used
			6	Timer 3 interrupt
			7	Interrupt bit
10	\$FF0A	65290		Interrupt masking
			0	Bit 8 of Raster compare (Register 11)
			1	Raster interrupt
			2	Light pen (Not possible with C-16)
			3	Timer 1 interrupt
			4	Timer 2 interrupt
			5	Not used
			6	Timer 3 interrupt
			7	Not used
11	\$FF0B	65291		Raster comparison (bit0-7)
12	\$FF0C	65292	0-1	Bit 8-9 of cursor position (Register 13)
			2-7	Not used
13	\$FF0D	65293		Hardware cursor position (bit0-7)
14	\$FF0E	65294		Frequency voice 1 (bit0-7)
15	\$FF0F	65295		Frequency voice 2 (bit0-7)
16	\$FF10	65296	0-1	Frequency voice 2 (bit8-9)
			2-7	Not used
17	\$FF11	65297	0-3	Volume (0=OFF, 15=LOUDEST)
			4	Switch voice 1 (1=ON)
			5	Switch voice 2 rectangular (1=ON)
			6	Switch voice 2 noise (1=ON)
			7	Sound reload bit
18	\$FF12	65298	0-1	Bit 8-9 frequency voice 1 (register 14)
			2	RAM/ROM bank (0=RAM, 1=ROM)
			3-5	Address of bitmap RAM (bit 13-15)
			6-7	Not used
19	\$FF13	65299	0	ROM bank status bit (read only)
			1	Force singleclock bit: 1=prohibit double time freq
			2-7	Address of character set (bit10-15)
20	\$FF14	65300	0-2	Not used
			3-7	Address of video RAM (bit11-15)
21	\$FF15	65301	0-3	Background 0 color
			4-6	Background 0 luminiscence
			7	Not used
22	\$FF16	65302	0-3	Background 1 color
			4-6	Background 1 luminiscence
			7	Not used
23	\$FF17	65303	0-3	Background 2 color
			4-6	Background 2 luminiscence
			7	Not used
24	\$FF18	65304	0-3	Background 3 color
			4-6	Background 3 luminiscence
			7	Not used
25	\$FF19	65305	0-3	Frame color
			4-6	Frame luminiscence
			7	Not used
26	\$FF1A	65306	0-1	Bit 8-9 of bit map reload (register 27)
			2-7	Not used
27	\$FF1B	65307		Bit map reload of character position (bit 0-7)
28	\$FF1C	65308	0	Bit 8 of raster-row (register 28)
			1-7	Not used
29	\$FF1D	65309		Current raster row (bit 0-7)
30	\$FF1E	65310		Current raster column (bit1-8)
31	\$FF1F	65311	0-2	Vertical sub address
			3-6	Flash rate
			7	Not used
62	\$FF3E	65342		ROM-select (write only)
63	\$FF3F	65343		RAM-select (write only)

## VIC AND SID CHIPS

VIC-CHIP ADDRESS \$D000-\$D02E (53248-53294)

ADDRESS	HEX	DECIMAL	BIT	DESCRIPTION
	\$D000	53248		Sprite 0 - X position (bits 0-8)
	\$D001	53249		Sprite 0 - Y position (bits 0-8)
	\$D002	53250		Sprite 1 - X position
	\$D003	53251		Sprite 1 - Y position
	\$D004	53252		Sprite 2 - X position
	\$D005	53253		Sprite 2 - Y position
	\$D006	53254		Sprite 3 - X position
	\$D007	53255		Sprite 3 - Y position
	\$D008	53256		Sprite 4 - X position
	\$D009	53257		Sprite 4 - Y position
	\$D00A	53258		Sprite 5 - X position
	\$D00B	53259		Sprite 5 - Y position
	\$D00C	53260		Sprite 6 - X position
	\$D00D	53261		Sprite 6 - Y position
	\$D00E	53262		Sprite 7 - X position
	\$D00F	53263		Sprite 7 - Y position



\$D010	53264	0	9th bit of sprite X co-ordinate
		1	Sprite 0
		2	Sprite 1
\$D011	53265	7	Sprite 2 etc through sprite 7
		6	VIC Control Register
		5	Raster compare register, Bit 9
		4	1-Enable extended colour text mode
		3	1-Enable bit map mode
		2-0	1-Blank screen to border
\$D012	53266		1-25 row text display, 0-24 row text display
\$D013	53267		Smooth scroll to Y dot position
\$D014	53268		Raster compare register, Position of raster on screen
\$D015	53269		Light pen X position
			Light pen Y position
		0	Enable or disable sprite
		1	1-Enable sprite 0
		2	1-Enable sprite 1
\$D016	53270		1-Enable sprite 2 etc through sprite 7
		4	VIC Control Register
		3	1-Multicolour mode on
		2-0	1-40 Column text: 0-39 column text
\$D017	53271		Smooth scroll to X position
		0	Sprite Vertical Expansion
		1	Expand sprite 0 Vertically
		2	Expand sprite 1 Vertically
\$D018	53272		Expand sprite 2 Vertically etc through to sprite 7
			VIC Memory Control
		7-4	Video matrix base address
\$D019	53273	3-0	Character set base address
			VIC Interrupt Flags
		7	Set to any VIC IRQ condition
		3	Light pen triggered (bit 7)
		2	Sprite vs sprite triggered (bit 7)
		1	Sprite vs background triggered (bit 7)
		0	Raster compare triggered (bit 7)
\$D01A	53274		VIC Interrupt Switches
		3	1-Enable light pen interrupt
		2	1-Sprite vs sprite enabled
		1	1-Sprite vs background enabled
		0	1-Raster compare enabled
\$D01B	53275		Sprite Priority Registers
		0	1-Sprite 0 passes in front of graphics
		1	1-Sprite 1 passed in front of graphics etc through sp 7
\$D01C	53276		Sprite multi-colour select
		0	1-Sprite 0 is multicolour
		1	1-Sprite 1 is multicolour etc through sprite 7
\$D01D	53277		Sprite Horizontal Expansion
		0	1-Sprite 0 expanded horizontally
		1	1-Sprite 1 expanded horizontally etc through sprite 7
\$D01E	53278		Sprite vs sprite collision detection. If a sprite is touching a sprite, the bit for that sprite is turned on.
\$D01F	53279		Sprite vs background. If sprite has hit text or background, relevant bit is set.
\$D020	53280		Border colour
\$D021	53281		Background colour
\$D022	53282		Multi-colour 1
\$D023	53283		Multi-colour 2
\$D024	53284		Multi-colour 3
\$D025	53285		Sprite multi-colour
\$D026	53286		Sprite multi-colour
\$D027	53287		Sprite 0 colour
\$D028	53288		Sprite 1 colour
\$D029	53289		Sprite 2 colour
\$D02A	53290		Sprite 3 colour
\$D02B	53291		Sprite 4 colour
\$D02C	53292		Sprite 5 colour
\$D02D	53293		Sprite 6 colour
\$D02E	53294		Sprite 7 colour
\$D030	53296	0	C128 ONLY 2MHz
			Determines if the C128 operates at 2MHz or 1MHz. If the bit is set then there is no access to the VIC chip and the C128 operates in the 2 Mhz mode.

SID CHIP ADDRESS \$D400-\$D41C (\$54272-\$54300)

KEY	DECIMAL	BIT	DESCRIPTION
\$D400	54272		Voice 1: low byte of frequency
\$D401	54273		Voice 1: High byte of frequency
\$D402	54274		Voice 1: Low byte of pulse width
\$D403	54275	3-0	Voice 1: High byte of pulse width
\$D404	54276		Voice 1 Control Register
		7	1-Random noise
		6	1-Pulse waveform on
		5	1-Sawtooth waveform on
		4	1-Triangle waveform on
		3	1-Disable voice 1
		2	1-Ring modulate voice 1 with voice 3
		1	1-Synchronize voice 1 with freq of voice 3
		0	1-Start attack,decay,sustain: 0=Start release
\$D405	54277		Voice 1 Attack/decay
		7-4	Attack cycle duration
\$D406	54278		Voice 1 Sustain/release
		3-0	Decay cycle duration
			Voice 2: low byte of frequency
\$D407	54279		Voice 2: high byte of frequency
\$D408	54280		Voice 2: low byte of pulse width
\$D409	54281		Voice 2: high byte of pulse width
\$D40A	54282	3-0	Voice 2 Control Register
\$D40B	54283		1-Random noise on
		6	1-Pulse waveform on
		5	1-Sawtooth waveform on
		4	1-Triangle waveform on
		3	1-Disable oscillator 1
		2	1-Ring modulate oscillator 2 with oscillator 1
		1	1-Synchronize oscillator 2 with oscillator 1 frequency
		0	1-Start attack,decay,sustain: 0=Start release
\$D40C	54284		Voice 2 Attack/decay
		7-4	Attack cycle duration
		3-0	Decay cycle duration

\$D40D	54285		Voice 2 Sustain/release
		7-4	Sustain cycle duration
		3-0	Release cycle duration
\$D40E	54286		Voice 3: low byte of frequency
\$D40F	54287		Voice 3: high byte of frequency
\$D410	54288		Voice 3: low byte of pulse width
\$D411	54289		Voice 3: high byte of pulse width
\$D412	54290		Voice 3 Control Register
		7	1-Random noise on
		6	1-Pulse waveform on
		5	1-Sawtooth waveform on
		4	1-Triangle waveform on
		3	1-Disable voice
		2	1-Ring modulate oscillator 3 with oscillator 2 output
		1	1-Synchronize oscillator 3 with freq of oscillator 2
		0	1-Start attack,decay,sustain: 0=Start release
\$D413	54291		Voice 3 Attack/decay
		7-4	Attack cycle duration
\$D414	54292		Voice 3 Sustain/release
		3-0	Decay cycle duration
			Voice 3 Sustain/release
		7-4	Sustain cycle duration
\$D415	54293		Release cycle duration
\$D416	54294	2-0	Filter cut-off low nybble
\$D417	54295		Filter cut-off high byte
			Filter Control
		7-4	Filter resonance
		3	1-External input to Filter
		2	1-Voice 3 to filter
		1	1-Voice 2 to filter
		0	1-Voice 1 to filter
\$D418	54296		Filter Volume And Mode
		7	1-Turn off voice 3 output
		6	1-High pass filter on
		5	1-Band pass filter on
		4	1-Low pass filter on
		3-0	Output volume
\$D419	54297		A/D convertor for paddle 1
\$D41A	54298		A/D convertor for paddle 2
\$D41B	54299		Produces random number when voice 3 set to noise
\$D41C	54300		Output of voice 3 envelope generator

6510 ADDRESSING MODES & OPCODES

The following table gives the HEX values for the various opcodes in their individual addressing modes. The following key to be used for the Address Mode

A = Accumulator  
# = Immediate  
ZP = Zero page  
AB = Absolute  
ABX = Absolute X  
ABY = Absolute Y  
ZPX = Zero page X  
ZPY = Zero page Y  
,X = Indexed X  
,Y = Indexed Y

mnemonic	ADDRESSING MODE									
	A	#	ZP	AB	ABX	ABY	ZPX	ZPY	,X	,Y
ADC	-	69	65	6D	7D	79	75	-	61	71
AND	-	29	25	2D	3D	39	35	-	21	31
ASL	0A	-	06	0E	1E	-	16	-	-	-
BIT	-	-	24	2C	-	-	-	-	-	-
CMF	-	C9	C5	CD	DD	D9	D5	-	C1	D1
CPX	-	E0	E4	EC	-	-	-	-	-	-
CPY	-	C0	C4	CC	-	-	-	-	-	-
DEC	-	-	C6	CE	DD	-	D6	-	-	-
EOR	-	49	45	4D	5D	59	55	-	41	51
INC	-	-	E6	EE	FD	-	F6	-	-	-
LDA	-	A9	A5	AD	BD	B9	B5	-	A1	B1
LDX	-	A2	A6	AE	-	BE	-	B6	-	-
LDY	-	A0	A4	AC	BC	-	B4	-	-	-
LSR	4A	-	46	4E	5E	-	56	-	-	-
ORA	-	09	05	0D	1D	19	15	-	01	11
ROL	2A	-	26	2E	3E	-	36	-	-	-
ROR	6A	-	66	6E	7E	-	76	-	-	-
SBC	-	E9	E5	ED	FD	F9	F5	-	E1	F1
STA	-	-	85	8D	9D	99	95	-	81	91
STX	-	-	86	8E	-	-	-	86	-	-
STY	-	-	84	8C	-	-	84	-	-	-

GROUPED INSTRUCTIONS

BRANCH INSTRUCTIONS

BPL	BMI	BVC	BVS	BCC	BCS	BNE	BEQ
10	30	50	70	90	B0	D0	F0

TRANSFER INSTRUCTIONS

JXA	JAX	JYA	JAY	JXS	JXA
BA	AA	9B	AB	BA	9A

STACK INSTRUCTIONS

PHP	PLP	PHA	PLA
0B	2B	4B	6B

JUMP INSTRUCTIONS

BRK	JSR	RTI	RTS	JMP	JMP	NOP
00	20	40	60	4C	6C	EA

FLAG INSTRUCTIONS

CLC	SEC	CLI	SEI	CLV	CLD	SED
1B	3B	5B	7B	8B	DB	FB

INC/DEC INSTRUCTIONS

DEY	INY	DEX	INX
8B	CB	CA	EB

## ASCII AND SCREEN CODES

SCREEN CODE	ASCII CODE	CHARACTER	SCREEN CODE	ASCII CODE	CHARACTER
HEX DECIMAL	HEX DECIMAL	SET1 SET2	HEX DECIMAL	HEX DECIMAL	SET1 SET2
\$00 0	\$40 64	@ @	\$40 64	\$60 96	A A
\$01 1	\$41 65	A a	\$41 65	\$61 97	B B
\$02 2	\$42 66	B b	\$42 66	\$62 98	C C
\$03 3	\$43 67	C c	\$43 67	\$63 99	D D
\$04 4	\$44 68	D d	\$44 68	\$64 100	E E
\$05 5	\$45 69	E e	\$45 69	\$65 101	F F
\$06 6	\$46 70	F f	\$46 70	\$66 102	G G
\$07 7	\$47 71	G g	\$47 71	\$67 103	H H
\$08 8	\$48 72	H h	\$48 72	\$68 104	I I
\$09 9	\$49 73	I i	\$49 73	\$69 105	J J
\$0A 10	\$4A 74	J j	\$4A 74	\$6A 106	K K
\$0B 11	\$4B 75	K k	\$4B 75	\$6B 107	L L
\$0C 12	\$4C 76	L l	\$4C 76	\$6C 108	M M
\$0D 13	\$4D 77	M m	\$4D 77	\$6D 109	N N
\$0E 14	\$4E 78	N n	\$4E 78	\$6E 110	O O
\$0F 15	\$4F 79	O o	\$4F 79	\$6F 111	P P
\$10 16	\$50 80	P p	\$50 80	\$70 112	Q Q
\$11 17	\$51 81	Q q	\$51 81	\$71 113	R R
\$12 18	\$52 82	R r	\$52 82	\$72 114	S S
\$13 19	\$53 83	S s	\$53 83	\$73 115	T T
\$14 20	\$54 84	T t	\$54 84	\$74 116	U U
\$15 21	\$55 85	U u	\$55 85	\$75 117	V V
\$16 22	\$56 86	V v	\$56 86	\$76 118	W W
\$17 23	\$57 87	W w	\$57 87	\$77 119	X X
\$18 24	\$58 88	X x	\$58 88	\$78 120	Y Y
\$19 25	\$59 89	Y y	\$59 89	\$79 121	Z Z
\$1A 26	\$5A 90	Z z	\$5A 90	\$7A 122	+ +
\$1B 27	\$5B 91	[ [	\$5B 91	\$7B 123	, ,
\$1C 28	\$5C 92	\ \	\$5C 92	\$7C 124	- -
\$1D 29	\$5D 93	] ]	\$5D 93	\$7D 125	. .
\$1E 30	\$5E 94	^ ^	\$5E 94	\$7E 126	_ _
\$1F 31	\$5F 95	~ ~	\$5F 95	\$7F 127	~ ~
\$20 32	\$60 96	SPC SPC	\$60 96	\$A0 160	SSPC SSPC
\$21 33	\$61 97	! !	\$61 97	\$A1 161	! !
\$22 34	\$62 98	@ @	\$62 98	\$A2 162	@ @
\$23 35	\$63 99	A A	\$63 99	\$A3 163	A A
\$24 36	\$64 100	B B	\$64 100	\$A4 164	B B
\$25 37	\$65 101	C C	\$65 101	\$A5 165	C C
\$26 38	\$66 102	D D	\$66 102	\$A6 166	D D
\$27 39	\$67 103	E E	\$67 103	\$A7 167	E E
\$28 40	\$68 104	F F	\$68 104	\$A8 168	F F
\$29 41	\$69 105	G G	\$69 105	\$A9 169	G G
\$2A 42	\$6A 106	H H	\$6A 106	\$AA 170	H H
\$2B 43	\$6B 107	I I	\$6B 107	\$AB 171	I I
\$2C 44	\$6C 108	J J	\$6C 108	\$AC 172	J J
\$2D 45	\$6D 109	K K	\$6D 109	\$AD 173	K K
\$2E 46	\$6E 110	L L	\$6E 110	\$AE 174	L L
\$2F 47	\$6F 111	M M	\$6F 111	\$AF 175	M M
\$30 48	\$70 112	N N	\$70 112	\$B0 176	N N
\$31 49	\$71 113	O O	\$71 113	\$B1 177	O O
\$32 50	\$72 114	P P	\$72 114	\$B2 178	P P
\$33 51	\$73 115	Q Q	\$73 115	\$B3 179	Q Q
\$34 52	\$74 116	R R	\$74 116	\$B4 180	R R
\$35 53	\$75 117	S S	\$75 117	\$B5 181	S S
\$36 54	\$76 118	T T	\$76 118	\$B6 182	T T
\$37 55	\$77 119	U U	\$77 119	\$B7 183	U U
\$38 56	\$78 120	V V	\$78 120	\$B8 184	V V
\$39 57	\$79 121	W W	\$79 121	\$B9 185	W W
\$3A 58	\$7A 122	X X	\$7A 122	\$BA 186	X X
\$3B 59	\$7B 123	Y Y	\$7B 123	\$BB 187	Y Y
\$3C 60	\$7C 124	Z Z	\$7C 124	\$BC 188	Z Z
\$3D 61	\$7D 125	+ +	\$7D 125	\$BD 189	+ +
\$3E 62	\$7E 126	, ,	\$7E 126	\$BE 190	, ,
\$3F 63	\$7F 127	- -	\$7F 127	\$BF 191	- -

## C64 POKES/PEEKs/TIPS

The following table gives some useful POKES, PEEKs, and other programming TIPS. The table is not laid out in any specific manner. These are just some of the bits and pieces that I have picked up over the last two years. The list is meant for BASIC programmers, but obviously, MACHINE CODE users can work out the M/C equivalents.

NOTE:- In all cases, LOW, HI and AD = Low byte of an address, High byte of an address and the address itself.

### POKE and/or PEEK

poke56334,peek(56334)and254  
 poke56334,peek(56334)or1  
 poke1,peek(1)and252  
 poke1,peek(1)or3  
 poke43,low:poke44,hi:peekad,0:clr  
 poke55,low:poke56,hi:clr  
 poke53265,peek(53265)or64  
 poke53265,peek(53265)and181  
 poke53270,peek(53270)or16  
 poke53270,peek(53270)and239  
 poke53272,(peek(53272)and241)orx  
 poke53272,(peek(53272)and241)or4  
 poke53272,(peek(53272)and241)or6  
 poke53272,(peek(53272)and15)orx  
 poke648,x  
 poke646,x  
 poke211,column:poke214,row:sya58372

### Effect of instruction

Switch off interrupts  
 Interrupts back on  
 Basic ROM switched out  
 Basic ROM switched back in  
 Raise start of BASIC  
 Lower end of BASIC  
 Enables extended background mode  
 Disables extended background mode  
 Enables multicolour mode  
 Disables multicolour mode  
 Activate character set designated by 'X'  
 Switch off character set (upper case)  
 Switch off character set (lower case)  
 Relocate video ram to add denoted by 'X'  
 Tell vid controller of new scan page no.  
 Change character color to that of 'X'  
 Place cursor at designated position

poke204,0  
 poke207,0:poke204,1  
 poke199,1  
 poke199,0  
 poke216,x  
 poke53265,peek(53265)and239  
 poke53265,peek(53265)or16  
 poke53265,59  
 poke788,52  
 poke788,49  
 poke792,193  
 poke792,71  
 poke808,234  
 poke808,128  
 poke808,64  
 poke808,0  
 poke808,x  
 poke198,0:wait198,1  
 poke775,1  
 poke775,167  
 poke775,131:poke775,164  
 poke801,0:poke802,0:poke818,165  
 poke818,131:poke819,164  
 poke818,157  
 poke818,165  
 poke22,35  
 poke22,25  
 printpeek(144)  
 printpeek(147)  
 printpeek(150)  
 printpeek(152)  
 printpeek(153)  
 printpeek(154)  
 printpeek(184)  
 printpeek(185)  
 printpeek(186)  
 sys64738  
 sys62255  
 sys65499  
 sys44808  
 sys42115

Turn on cursor  
 Turn off cursor  
 Reverse video on  
 Reverse video off  
 Insert mode on 'X' denotes no of inserts  
 Switch screen off  
 Switch screen on  
 Switch on Hi-res mode  
 Disable run/stop key  
 Re-enable run/stop key  
 Disable restore key  
 Re-enable restore key  
 Disable BREAK and LIST  
 All keys repeat  
 Disable repeat key function  
 Normal key repeat condition  
 x-signifies delay of repeat(64'ths sec)  
 Clear keyboard buff & wait for keypress  
 Disable list  
 Restore list  
 Disable list (alt method)  
 Disable save command  
 Disable save command (alt method)  
 Disable load command  
 Re-enable load command  
 Remove line numbers from basic prg  
 Replace line numbers  
 Returns value of ST  
 Returns LOAD/VERIFY flag value  
 Cassette motor flag status  
 Number of files open  
 Actual input device  
 Actual output device  
 Actual file  
 Actual secondary address  
 Actual device  
 Activate system re-set  
 Close all open files  
 Re-sets TI to 000000  
 Gives invisible 'Syntax Error'  
 Ends program without the READY prompt

## HEX TO DECIMAL CONVERTER

HEX	DECIMAL	HEX	DECIMAL	HEX	DECIMAL	HEX	DECIMAL
LOW	HIGH	LOW	HIGH	LOW	HIGH	LOW	HIGH
\$00 0	0	\$40 64	16384	\$80 128	32768	\$C0 192	49152
\$01 1	256	\$41 65	16640	\$81 129	33024	\$C1 193	49408
\$02 2	512	\$42 66	16896	\$82 130	33280	\$C2 194	49664
\$03 3	768	\$43 67	17152	\$83 131	33536	\$C3 195	49920
\$04 4	1024	\$44 68	17408	\$84 132	33792	\$C4 196	50176
\$05 5	1280	\$45 69	17664	\$85 133	34048	\$C5 197	50432
\$06 6	1536	\$46 70	17920	\$86 134	34304	\$C6 198	50688
\$07 7	1792	\$47 71	18176	\$87 135	34560	\$C7 199	50944
\$08 8	2048	\$48 72	18432	\$88 136	34816	\$C8 200	51200
\$09 9	2304	\$49 73	18688	\$89 137	35072	\$C9 201	51456
\$0A 10	2560	\$4A 74	18944	\$8A 138	35328	\$CA 202	51712
\$0B 11	2816	\$4B 75	19200	\$8B 139	35584	\$CB 203	51968
\$0C 12	3072	\$4C 76	19456	\$8C 140	35840	\$CC 204	52224
\$0D 13	3328	\$4D 77	19712	\$8D 141	36096	\$CD 205	52480
\$0E 14	3584	\$4E 78	19968	\$8E 142	36352	\$CE 206	52736
\$0F 15	3840	\$4F 79	20224	\$8F 143	36608	\$CF 207	52992
\$10 16	4096	\$50 80	20480	\$90 144	36864	\$D0 208	53248
\$11 17	4352	\$51 81	20736	\$91 145	37120	\$D1 209	53504
\$12 18	4608	\$52 82	20992	\$92 146	37376	\$D2 210	53760
\$13 19	4864	\$53 83	21248	\$93 147	37632	\$D3 211	54016
\$14 20	5120	\$54 84	21504	\$94 148	37888	\$D4 212	54272
\$15 21	5376	\$55 85	21760	\$95 149	38144	\$D5 213	54528
\$16 22	5632	\$56 86	22016	\$96 150	38400	\$D6 214	54784
\$17 23	5888	\$57 87	22272	\$97 151	38656	\$D7 215	55040
\$18 24	6144	\$58 88	22528	\$98 152	38912	\$D8 216	55296
\$19 25	6400	\$59 89	22784	\$99 153	39168	\$D9 217	55552
\$1A 26	6656	\$5A 90	23040	\$9A 154	39424	\$DA 218	55808
\$1B 27	6912	\$5B 91	23296	\$9B 155	39680	\$DB 219	56064
\$1C 28	7168	\$5C 92	23552	\$9C 156	39936	\$DC 220	56320
\$1D 29	7424	\$5D 93	23808	\$9D 157	40192	\$DD 221	56576
\$1E 30	7680	\$5E 94	24064	\$9E 158	40448	\$DE 222	56832
\$1F 31	7936	\$5F 95	24320	\$9F 159	40704	\$DF 223	57088
\$20 32	8192	\$60 96	24576	\$A0 160	40960	\$E0 224	57344
\$21 33	8448	\$61 97	24832	\$A1 161	41216	\$E1 225	57600
\$22 34	8704	\$62 98	25088	\$A2 162	41472	\$E2 226	57856
\$23 35	8960	\$63 99	25344	\$A3 163	41728	\$E3 227	58112
\$24 36	9216	\$64 100	25600	\$A4 164	41984	\$E4 228	58368
\$25 37	9472	\$65 101	25856	\$A5 165	42240	\$E5 229	58624
\$26 38	9728	\$66 102	26112	\$A6 166	42496	\$E6 230	58880
\$27 39	9984	\$67 103	26368	\$A7 167	42752	\$E7 231	59136
\$28 40	10240	\$68 104	26624	\$A8 168	43008	\$E8 232	59392
\$29 41	10496	\$69 105	26880	\$A9 169	43264	\$E9 233	59648
\$2A 42	10752	\$6A 106	27136	\$AA 170	43520	\$EA 234	59904
\$2B 43	11008	\$6B 107	27392	\$AB 171	43776	\$EB 235	60160
\$2C 44	11264	\$6C 108	27648	\$AC 172	44032	\$EC 236	60416
\$2D 45	11520	\$6D 109	27904	\$AD 173	44288	\$ED 237	60672
\$2E 46	11776	\$6E 110	28160	\$AE 174	44544	\$EE 238	60928
\$2F 47	12032	\$6F 111	28416	\$AF 175	44800	\$EF 239	61184
\$30 48	12288	\$70 112	28672	\$B0 176	45056	\$F0 240	61440
\$31 49	12544	\$71 113	28928	\$B1 177	45312	\$F1 241	61696
\$32 50	12800	\$72 114	29184	\$B2 178	45568	\$F2 242	61952
\$33 51	13056	\$73 115	29440	\$B3 179	45824	\$F3 243	62208
\$34 52	13312	\$74 116	29696	\$B4 180	46080	\$F4 244	62464
\$35 53	13568	\$75 117	29952	\$B5 181	46336	\$F5 245	62720
\$36 54	13824	\$76 118	30208	\$B6 182	46592	\$F6 246	62976
\$37 55	14080	\$77 119	30464	\$B7 183	46848	\$F7 247	63232
\$38 56	14336	\$78 120	30720	\$B8 184	47104	\$F8 248	63488
\$39 57	14592	\$79 121	30976	\$B9 185	47360	\$F9 249	63744
\$3A 58	14848	\$7A 122	31232	\$BA 186	47616	\$FA 250	64000
\$3B 59	15104	\$7B 123	31488	\$BB 187	47872	\$FB 251	64256
\$3C 60	15360	\$7C 124	31744	\$BC 188	48128	\$FC 252	64512
\$3D 61	15616	\$7D 125	32000	\$BD 189	48384	\$FD 253	64768
\$3E 62	15872	\$7E 126	32256	\$BE 190	48640	\$FE 254	65024
\$3F 63	16128	\$7F 127	32512	\$BF 191	48896	\$FF 255	65280



# Foreign Formats

*In theory, the C128 has the valuable ability to read a wide range of CP/M disk formats. In practice, it isn't so easy. Your Commodore shows you how to do it.*

One of the most interesting features of the C-128's CP/M mode is its ability to read a wide number of MFM disk formats created on other CP/M systems. The feature was only provided so that the 128 owner would have ready access to the full range of CP/M applications without having to rely on companies to copy programs on to the non-standard Commodore format diskette. By contrast owners of, for example, Amstrad machines are in the main limited to a selection of the most popular CP/M packages.

While this in itself is a major benefit, provision of the facility also provides some less well-publicised advantages. Of course it means that you can now create your own programs and datafiles for use on other CP/M machines. What, however will be of interest to a greater number of 128 owners is that the disk drives work faster with MFM disks than with the standard GCR format. The main reason for this is almost certainly, because the physical track/sector layout of Commodore disks does not fit very well with CP/M's internal logical representation of a disk. Additionally, the new disk ROM routines for handling MFM disks are the only tide sections within the disk ROMs.

## Horror movie

At the point it would be very easy to digress into a discussion of CP/M internals and the horror movie lurking within your disk drive. For readers interested in these topics I have suggested books on both topics at the end of this

article and we will just consider the more immediate problem of creating an MFM format disk.

Further investigations have revealed that while Commodore originally intended at one time to provide an MFM formatting facility from within CP/M, it is now extremely difficult to do so. Fortunately, however, it is relatively simple to do this using BASIC 7.0.

In order to implement CP/M, the new disk drives support a set of instructions called Burst Commands. These account for the slight improvement in disk performance that is always obtained in CP/M mode by using faster data transfers. The main functions in this group are for fast read and write of 128 byte CP/M logical records and a special fast program LOAD command. Also included within the set is a general purpose MFM formatting command. The information on this is found on page 84 of the 1571 Disk Drive Users Guide. It isn't too hard to use Burst Commands — I successfully formatted a disk to KAYPRO II format at the second attempt.

## Straightforward

Everything seemed quite straightforward, so I was somewhat surprised when every other format I attempted to create was greeted by CP/M with the response MISSING. This is shorthand for: "I have searched through my internal tables, oh master, and cannot find an entry that matches this format". Clearly another gem from the Commodore School of Techni-

cal Authorship. The disk manual seems to be accurate, but gives no details on the actual formats supported. They are given in the CP/M sections of the main manual, but this does not tell you how the sectors are numbered, which you need to identify.

There is one place where you can find this information and you will only have it if you have bought the CP/M utilities and documentation pack. This package also contains a disk of CP/M sources and at the end of the BIOS file CXDISK. ASM you will find Disk Parameter Block (DPB) Table. The DPB Table holds preset information for a range of different file formats, but is modifiable by the user.

After the DPB entries for the Commodore formats and the MFM formats there are a number of blank entries and two unimplemented ones used on Morrow machines. This means it is possible to insert new entries without having to discard any of the existing ones. The easiest way to modify this file is to use a word-processor, otherwise you will have to struggle with the infamous ED editor.

The DPB table is unique to Commodore, a most CP/M machines have only a single DPB. This is required by the operating system in order to convert from the logical structure of the disk used in the BDOS section and the physical structure of tracks and sectors on the disk. Logically, CP/M considers a disk to contain a number of 128 byte records, which are organised into blocks of between 1k and 16k bytes in length. For the C128 the block size is 1k for single-sided disks and 2k for double-sided disks.

By Paul Schofield



This is an important parameter as it defines the minimum size of a CP/M disk file, regardless of how little data it may contain.

The parameters in the DPB are used to calculate the numbers of the sectors on a particular track that correspond to a logical block. For formatting purposes the only parameters of interest are the sector size, the number of sectors per track and the number allocated to the first sector on a track. The other parameters concern the order in which the disk drive writes sectors on the disk. These parameters can be specified in the format command, but they should not be required.

The demonstration program shown in Listing 1 will create two of the most useful formats. The KAYPRO II format packs more data than normal on a disk and the IBM-8 formats are very widely used. Using the table of parameters for other formats provided in Table 1, it would of course be possible to extend the program to allow the creation of all the supported formats. However, unless you are planning to provide a CP/M disk copying service, it is probably better to limit the formats you are using.

### Further reading

As mentioned earlier, here are a couple of book recommendations for readers who would like to pursue these subjects further. A very good introduction to this subject is provided by *CP/M The Software Bus*. This is a programmer's companion written by Andrew Clarke, Mike Eaton and David Powys-Lybbe and published by Sigma Technical Press. This book scores high on readability and follows a sensible progression from the very basics through editors, assemblers and compilers to the operating system internals.

I have only two reservations in recommending this to C-128 users. The first is that the book concentrates on CP/M rather than CP/M Plus which is used on the 128. While the differences are fully covered but not in great detail. Secondly in my copy at least (several years old) the chapter on CP/M programming languages badly needs updating and does not even mention some of the best compilers available. In all other respects this book should suit all but

MFM Formatting Parameters Table

No.	Format	Sides	Sct size (B5)	Sct/Trk (B7)	1st Sct (B3)
1	Epson QX10	2	1	16	129
2	Epson QX	2	2	10	129
3	IBM-8 SS	1	2	8	129
4	IBM-8 DS	2	2	8	129
5	KAYPRO IV	2	2	10	128
6	KAYPRO II	1	2	10	128
7	Osborne DD	2	3	5	129
8	Osborne SD	1	3	5	129
9	Epson Euro	2	1	16	129

Formats 10-16 are available for user definition.

#### PROGRAM: MFM FORMATTER

```

10 REM 1570 MFM DISK FORMATTER -
  BY PAUL SCHOFIELD
20 SCNCLR:CHAR 1,30,3,"MFM DISK
  FORMATTER"
30 CHAR 1,30,4,"--- ----"
40 CHAR 1,10,7,"1. KAYPRO II"
50 CHAR 1,10,9,"2. IBM-8 SS"
60 CHAR 1,10,11,"PLEASE ENTER RE
  QUIRED FORMAT : "
70 CHAR 1,41,11,"":INPUT F:IF F>
  2 OR F<1 THEN 60
80 CHAR 1,10,14,"INSERT DISK AND
  ENTER DEVICE NUMBER (8-11) :
  "
90 CHAR 1,56,14,"":INPUTD:IFD<8
  OR D>11 THEN 80
100 IF F=1 THEN B5=2:B7=10:B3=12
  8
110 IF F=2 THEN B5=2:B7=8:B3=129
120 C$="UO"+CHR$(6)+CHR$(B3)+CHR
  $(0)+CHR$(B5)+CHR$(39)+CHR$(B7)
130 CHAR 1,10,16,"":OPEN 1,D,15,
  C$:PRINT#1,"UO"+CHR$(4):GET#1,A$
  :CLOSE1
140 SB=ASC(A$)-48:IF DS>1 OR SB<
  >0 THEN PRINT "FORMAT ERROR - ";
  DS$;"      STATUS = ";SB:END
150 PRINT "ANOTHER DISK (Y/N) :
  "
160 INPUT Y$:IF Y$="Y" OR Y$="C$
  Y)" THEN CHAR 1,10,16,"
  "
  :GOTO 130:EL
SE SCNCLR
170 REM FOR 1571 DOUBLE SIDED FO
  RMATS CHANGE CHR$(6) TO CHR$(38)
  IN LINE 110

```

the most dedicated hackers and is quite reasonably priced.

By contrast the 1570/71 disk drives are still too new to have had many books written about them yet. I know of only 3, which are in fact editions of the same book in German, American and English. The English version is entitled *The Anatomy of the 1571 Disk Drive* and is published by First Publishing. If this book had appeared a little later (the German original has been available almost as long as the drives) it would probably have been excellent. As it is it is simply very good. The main weakness is that in places they have had to anticipate what information users would require and so have not included everything that Commodore forgot. All the disk housekeeping commands, file types, and special user commands are fully covered, although the inexperienced programmer would probably have appreciated more example programs.

This is not, however, a book aimed at the novice and over half of it is ROM listings. In this case the authors have done a really first class job of adding comments to these and they are genuinely useful. Despite this it is still difficult to follow some commands from reception to completion, which is hardly the fault of the authors, but is inherent in the byzantine structure of the ROMs. There is great potential for getting these C128 disk drives to perform better than intended and this is the type of book you need if you want to try.







19 2050 OPEN4,4	3A 3005 PRINT"[RIGHT2]COPIES OF	95 3130 CLOSE4:NEXTC:RETURN
FC 2060 C\$="1[SPC14]"	A SMALL SCREEN MATIX"	C9 3997 REM*****
EA 2070 D\$="2 6 3 1[SPC8]"	7F 3010 PRINTN\$;LEFT\$(O\$,15)LEF	*****
44 2080 E\$="8 4 2 6 8 4 2 1"	T\$(P\$,2);	B6 3998 REM LARGE SCREEN MATRIX
94 2089 REM** (SHIFT + O SHIFT +	A8 3015 INPUT"[SH]OW MANY COPIE	*
P)*8	S ";NR	0B 3999 REM*****
E5 2090 A\$="[SO,SP,SO,SP,SO,SP,	C2 3020 IFNR=0THEN3000	*****
SO,SP,SO,SP,SO,SP,SO,SP,SO,S	66 3030 FORC=1TONR	8D 4000 PRINTM\$;LEFT\$(O\$,5)LEFT
P]"	E6 3050 OPEN4,4	\$(P\$,2);"[ST]HIS ROUTINE WIL
4F 2099 REM** (SHIFT + L SHIFT +	2A 3069 REM** (LOGO + A)*1 (LOGO	L PRINT OUT A COPY OR"
@)*8	+ R)*21	DC 4005 PRINT"[RIGHT2]COPIES OF
20 2100 B\$="[SL,S@,SL,S@,SL,S@,	C0 3070 A\$="[CA,CR21]"	A LARGE SCREEN MATIX"
SL,S@,SL,S@,SL,S@,SL,S@,SL,S	30 3071 :	73 4010 PRINTN\$;LEFT\$(O\$,15)LEF
@]"	6A 3074 REM** (LOGO + R)*18 (LOG	T\$(P\$,2);
7B 2110 G\$="[C@4] [C@4] [C@4]"	O + S)*1	F4 4015 INPUT"[SH]OW MANY COPIE
3B 2120 PRINT#4,C\$ " "C\$ " "C\$	A9 3075 B\$="[CR18,CS]"	S ";NR
86 2130 PRINT#4,D\$ "[SSPC]"D\$ " "	2A 3076 :	69 4020 IFNR=0THEN4000
D\$	B9 3079 REM** (LOGO + Q)*1 (SHIF	4A 4030 FORC=1TONR
CF 2140 PRINT#4,E\$ "[SSPC]"E\$ "[S	T AND +)*21	D2 4050 OPEN4,4
SPC]"E\$ " DATA"	39 3080 C\$="[CQ,S+21]"	CB 4069 REM** (SHIFT + O SHIFT +
61 2150 FORL=1TO21	2F 3081 :	P)*8
09 2160 PRINT#4,AA\$A\$A\$A\$A\$ZZ\$	B8 3084 REM** (SHIFT AND +)*18 (	2E 4070 A\$="[SO,SP,SO,SP,SO,SP,
BB 2170 PRINT#4,AA\$B\$B\$B\$ "G\$Z	LOGO + W)*1	SO,SP,SO,SP,SO,SP,SO,SP,SO,S
Z\$	1C 3085 D\$="[S+18,CW]"	P]"
1B 2180 NEXTL:PRINT#4,AA\$:CLOSE	20 3086 :	60 4079 REM** (SHIFT + L SHIFT +
4:NEXTC	DD 3089 REM** (LOGO + Z)*1 (LOGO	@)*8
18 2190 RETURN	+ E)*21	67 4080 B\$="[SL,S@,SL,S@,SL,S@,
94 2300 STOP	D1 3090 E\$="[CZ,CE21]"	SL,S@,SL,S@,SL,S@,SL,S@,SL,S
25 2997 REM*****	25 3091 :	@]"
*****	D0 3094 REM** (LOGO + E)*18 (LOG	50 4090 FORL=1TO25
64 2998 REM SMALL SCREEN MATRIX	O + X)*1	1D 4100 PRINT#4,AA\$A\$A\$A\$A\$A\$ZZ
*	42 3095 F\$="[CE18,CX]"	\$
67 2999 REM*****	1E 3096 :	74 4110 PRINT#4,AA\$B\$B\$B\$B\$B\$ZZ
*****	DC 3100 PRINT#4,AA\$A\$;B\$ZZ\$	\$
61 3000 PRINTM\$;LEFT\$(O\$,5)LEFT	FE 3110 FORL=1TO24:PRINT#4,AA\$C	C3 4120 NEXTL:PRINT#4,AA\$:PRINT
\$(P\$,2);"[ST]HIS ROUTINE WIL	\$;D\$ZZ\$:NEXTL	#4:PRINT#4:CLOSE4:NEXTC:RETU
L PRINT OUT A COPY OR"	C8 3120 PRINT#4,AA\$E\$;F\$ZZ\$	RN

## Continued from page 42

70 First line of program when recalled later.

On cassette the files would have to be in the order:

- 1: PROG 1
- 2: SWAPPER 64 (machine code not Basic loader.)
- 3: PROG2

On disk the order does not matter.

When the store part instruction is used, the bottom of Basic is raised to the address of the beginning line number. This will cause no problems if used in program mode, but in direct mode it may be noticed. In such cases it is advisable to SAVE the program, reset the computer, and re-LOAD the program. This procedure is not required if the beginning line number was the first line of the program. It is advised that you only use this command in program mode.

## Extremely fast

Here is a short Basic program which will demonstrate the speed of SWAPPER 64.Type:

```
NEW
10 PRINT "HELLO"
20 SYS 49152,4,1
SYS 49152,2
RUN
```

The program scrolls 'HELLO' up the screen. What is happening is that line 10 is executed thus printing 'HELLO'. Then the program is swapped for that in storage which auto-RUNs. Since this is the same program, the above process is repeated.

## Getting it all in

The program is presented here in the form of a Basic loader. Type the program in using the SYNTAX CHECKER program that can be found on the LISTINGS page.

SAVE the program before you RUN it. On RUNNING you will be informed of any errors that you may have missed. When the program has been RUN successfully, type:

```
POKE43,0:POKE44,192:POKE45,35:
POKE46,195
SAVE "SWAPPER",1 (or ,8 for disk).
This SAVES a working copy of the machine code.
When the program is to be used, simply type the following command:
```

```
LOAD "SWAPPER",1,1 for tape or,
LOAD "SWAPPER",8,1 for disk.
```

If loaded directly (ie. as above) rather than from within a program then you must type NEW or CLR to reset the pointers after LOADING SWAPPER.

For those interested the machine code for SWAPPER is located between memory locations 49152 (\$C000) and 49991 (\$C347).



# Program Lock

*Are you keen to keep prying eyes off your beautiful Basic coding? This utility will help you to keep a few secrets.*

**H**ere is a program to make it very difficult for prying eyes to look at your Basic programs without your permission. PROG-LOCK is a 'skeleton' program which sits below your own program, so that when the program is loaded and listed, the result will be the message 'I0 SYS2076' which is far from informative.

To see the Basic program, you must RUN the program, which will then ask you for a password, which if entered correctly will allow you to run or list the Basic program. If you don't give the correct password, the machine resets itself!

## Getting Going

Type in the program and SAVE it as LOCK LOAD **before** RUNning it. If all the data is correct, PROG-LOCK is SAVED to tape or disk, as you require.

Now to keep out the nosey-parkers.

Turn the computer on and off, then LOAD PROG-LOCK back in. Type:

POKE43,124

followed by RETURN.

Now LOAD in the BASIC program you want to protect, and type:

POKE43,1

followed by RETURN.

The default password is set as ELEPHANT, but you change it by typing:

SYS2150

followed by RETURN, and you can then type is a new password of up to 20

characters (counting RETURN at the end as one character).

That's it! The program is now protected, and just needs to be SAVED.

Note that PROG-LOCK doesn't copy-protect your program, and would be rapidly defeated by an experienced hacker, but it will make the casual program thief think twice, not to mention the *cachet* it will give to your programs!

### PROGRAM: PROG LOCK

```

8B 10 X=16385:T=0:N=0
39 20 READA:IFA=-1THEN40
88 30 POKEX,A:T=T+A:N=N+1:X=X+1
    :GOTO20
ED 40 IFT<>10932THENPRINT"DATA
    VALUE ERROR":END
OF 50 IFN<>125THENPRINT"DATA AM
    OUNT ERROR":END
52 60 PRINT"(T)APE OF (D)ISK ?"
C7 70 GETD$:IFD$<>"T"ANDD$<>"D"
    THEN70
39 80 POKE251,1:IFD$="D"THENPOK
    E251,8
5C 90 POKE43,1:POKE44,64:POKE45
    ,127:POKE46,64:SAVE"PROG-LOC
    K".PEEK(251)
1D 1000 DATA 11,8,10,0,158,50,4
    8,55,54,0
  
```

```

FB 1010 DATA 0,0,147,8,14,32,20
    8,65,83,83
A6 1020 DATA 87,79,82,68,32,63,
    0,169,13,160
12 1030 DATA 8,32,30,171,160,0,
    32,207,255,153
3F 1040 DATA 64,3,200,201,13,20
    8,245,169,0,153
A7 1050 DATA 64,3,160,0,185,64,
    3,240,12,217
A6 1060 DATA 81,8,240,3,76,226,
    252,200,76,55
90 1070 DATA 8,169,124,133,43,1
    69,8,133,44,96
81 1080 DATA 69,76,69,80,72,65,
    78,84,13,0
3D 1090 DATA 153,34,72,69,76,76
    ,79,34,0,108
BE 1100 DATA 0,169,13,160,8,32,
    30,171,160,0
72 1110 DATA 32,207,255,153,81,
    8,200,201,13,208
95 1120 DATA 245,96,0,0,0,-1
  
```



**Want to  
use the  
programmes  
in this  
publication?**

**T**yping in long programs can be a pretty daunting task. Once you've entered the program there will no doubt be typing errors that need to be corrected. Why not save yourself time and trouble by buying a disk or cassette of the programme from this publication. All of the programmes that are presented here are on the cassette or disk at a bargain price of £6.00 for disk or £4.00 for cassette.

The disk and cassette are only available mail order from the address on the order form. A cheque payable to ASP Ltd for the correct amount should be included with the order. Overseas customers should add £1.00 for postage.

**Can't afford  
the time to  
type them in?**

**Why not buy  
them all  
on disk  
or cassette?**

ORDER FORM — PLEASE COMPLETE IN BLOCK CAPITALS

NAME	QUANTITY	PRICE	ORDERCODE	TOTAL
SERIOUS USER DISK		£6.00	YSUGD	
SERIOUS USER TAPE		£4.00	YSUGC	
OVERSEAS POSTAGE		£1.00		
			TOTAL	

NAME .....

ADDRESS .....

..... POSTCODE.....

I enclose a cheque/postal order for £..... made payable to ASP LTD. for the Your Commodore Serious User Guide Disk/Tape.

All orders should be sent to: Your Commodore, Readers Services, Argus Specialist Publications, 9 Hall Road, Hemel Hempstead, Herts HP2 7BH.

Please allow 28 days for delivery.



# AT LAST!

## AN ECONOMICAL ALTERNATIVE TO THE BULKY EXTERNAL AMIGA DISK DRIVES

### 3.5" EXTERNAL FLOPPY DISK DRIVE FOR THE COMMODORE AMIGA



#### CUMANA CAX 354

Amiga owners can now easily upgrade to twin floppy operation with the purchase of Cumana's high quality external 3.5 inch floppy drive. The Cumana CAX 354 conveniently takes its power from the host computer and offers a full 880K of formatted storage to either A500 owners or users of system 1 and 3 A1000 series -

- High quality NEC 3.5 inch double sided drive mechanism
- 1Mb Unformatted storage capacity
- High Reliability
- Fast Access

- Quiet operation
- Lower power consumption
- Connector enables easy addition of 5.25" drives

#### SPECIFICATIONS

Seek time (track to track) 3ms • Settling time 15ms • Rotational Speed 300 RPM • Data Transfer Rate 125/250 Kb per sec • Number of tracks 80 • Number of sides 2

### FED UP WITH PAYING HIGH PRICES FOR YOUR 5.25" FLOPPY DISKS???

#### JUST LOOK AT OUR PRICES!!!!



DS/DD 5.25" DISKS

AT THE SILLY

PRICE OF JUST £6.00 PER TEN

SAVE EVEN MORE MONEY

BUY TWO PACKS AND SAVE

ANOTHER £2.00

TWO PACKS OF TEN 5.25" DISKS

JUST £10.00

Complete with labels and write protect tabs.  
Prices include VAT and UK P&P.  
No fancy boxes to throw away. You get the highest quality disk at the lowest of prices.

### COMMODORE CABLES

#### CPC/1 CENTRONICS PRINTER CABLE

Commodore C64/128 user port to centronics printer cable. The cable is fitted with a line feed switch for the Epson range of printers. Works with all well known centronics printers. **ONLY £15.00 incl.**

#### CPC/2 SERIAL EXTENSION CABLES

Extend your commodore printer or disk drive cable by up to 2 metres  
1 Metre extension cable.....£5.00 incl  
2 Metre extension cable.....£7.00 incl

#### CPC/3 128D KEYBOARD EXTENSION

Do you find yourself restricted by the short keyboard cable on the C128D. Solve your problem with our 1 metre extension cable.

**We have mounted our C128 under the desk to save room.**  
**SPECIAL OFFER PRICE ONLY £15.00 incl**

### LOCKABLE DISK BOXES

#### DB3/90

3.5" disk box holds 90 disks first class value at only £12.50 or only £11.50 when you buy 10 or more 3.5" disks.

#### DB5/70

5.25" disk box holds 70 disks great value only £9.50 or only £8.50 when you buy 20 or more 5.25" disks.

#### DB5/100

5.25" disk box holds 100 disks bargain at only £11.50 or only £10.50 when you buy 20 or more 5.25" disks.

### DISK NIBBLER

Use both sides of your disks. Save the cost of the Nibbler with just one box of disks even at our prices. Only £5.00 or FREE if you buy 50 or more 5.25" disks.

#### SPECIAL OFFER

50 5.25" disks £25.00  
1 DB5/100 disk box £10.50  
Disk Nibbler F.O.C.

OUR NORMAL PRICE £46.50

OFFER PRICE £35.00

SAVE !!!!! £11.50

Prices include VAT and UK postage



### AT LAST!!!! 3.5" DISKS AT SENSIBLE PRICES

Double sided, double density 3.5" verbatim disks

**ONLY £16.00 for pack of ten disks**

**SAVE EVEN MORE MONEY!!!!**

**BUY TWO PACKS FOR ONLY £30.00**

These are not cheap disks but best quality disks at low prices.

**NASHA DS/DD 3.5" DISKS**

**BOXED, WITH LABEL**

**OUR LOW PRICE £23.00 per box ten.**

**SAVE EVEN MORE MONEY!!!**

**BUY TWO BOXES FOR ONLY £44.00**

We believe our prices are the lowest you will find.  
All prices include VAT and UK postage.

**NEW!**

**NEW!**

**NEW!**

**NEW!**

**NEW!**

### COMMODORE C64, C128 RS232 INTERFACE

AT LAST!! RS232 Interface that will not cost you the earth.

The H&P Computers Commodore RS232 Interface is a full industrial standard RS232 Interface with all handshaking lines, that plugs into the user port.

Will fit all modems and printers with a 25 way D connector.

Up till now you would have had to pay between £35.00 and £50.00 for a RS232 Interface for the C64/C128.

The H&P Computers RS232 is only £25.00 incl. and we even give you an xmodem comms program on disk free of charge.

**ONCE AGAIN WE BRING THE BEST FOR LESS.**  
**ONLY £25.00 INCL.**

**H&P COMPUTERS UK,**

**9 HORNBEAM WALK, WITHAM, ESSEX CM8 2SZ. Tel: (0376) 511471**



# TOTAL BACK-UP POWER CBM 64/128

## PERIPHERALS..THE FINAL FRONTIER..OUR MISSION..TO BOLDLY GO WHERE NO OTHER UTILITIES HAVE GONE BEFORE

CAPTAIN'S LOG...THE TOTAL SOLUTION TO ALL YOUR BACK-UP NEEDS... THE ULTIMATE BACK-UP CARTRIDGE HERE NOW !!

### REPORT ON FINDINGS

Action Replay Mk III is more powerful, more friendly and will back up more programs than any competing utility by taking a 'Snapshot' of the program in memory so it doesn't matter how it was loaded... from disk or tape, at normal or turbo speeds... the results are the same - Perfect!! Amazing!!!

### STARBASE UPDATE

- Simple to use: just press the button and make a complete backup: Tape to Tape, Tape to Disk, Disk to Tape. - THE PROCESS IS AUTOMATIC - JUST GIVE THE BACKUP A NAME.
- All backups will reload at turbo speed independently of the cartridge.
- Dual speed tape turbo system. Programs can load up to 3 times faster than commercial turbos - that's over 10 times normal Commodore speed.

- Freeze the action then view the program with the monitor feature. Add pokes for infinite lives etc. Then restart the game or backup - ideal for customised versions of your games.
- Picture Save. Save any multi-colour. Hires screen to disk or tape. Compatible with Blazing Paddles, Koala, Slideshow etc.
- Fully compatible with 1541, 1541C, 1570, 1571, and ehancer or any CBM compatible data recorder.
- For C64, 64C, 128, 128D (in 64 mode).
- Unique Sprite Monitor. Freeze the Action and view all the Sprites, watch the animations scroll across the screen. Save Sprites to disk or tape. Customise your games by loading sprites from one game to another - then restart the program or make a backup.

- Compatible with fast DOS and Turbo ROM systems.
- Backup process in turbo speed - faster than any rivals.
- Special compacting techniques. Each program is saved as a single file.
- Transfers multistage tape programs to disk - more than any other cartridge - even the extra stages are turbo load - a unique feature.
- Sprite Killer! make yourself indestructible by disabling Sprite collisions in games.
- Fast disk format (20 secs).
- Built-in unstoppable reset button.

**ONLY  
£29.99  
POST FREE**

**ACTION  
REPLAY  
MK III**

**PLUS Built In  
FASTLOADER**

Action Replay III even has a built in disk fast loader which speeds up loading 5 times. Uses no memory - invisible to the system. You could pay £20 alone for this feature.

**WARP 25  
BREAKS THROUGH  
THE 10 SECOND BARRIER!**

Action Replay III now comes with an amazing new Disk Bootloader that will reload your backups at TWENTY FIVE TIMES normal speed. The fastest disk turbo yet devised!! There are NO CATCHES. WARP 25 works with ALL your games. Works with any disk drive. No pre-load required - No hardware modifications necessary - No user knowledge required - programs load INDEPENDENTLY. LOADING TIME - 9.8 SECONDS (for a typical game saved by WARP 25 in conjunction with ACTION REPLAY III). This time is for the COMPLETE load process from start to finish. Reload is entirely INDEPENDENT of the cartridge or any other hardware. Compare these (accurate!) figures for some rival backup systems:

SYSTEM	LOAD TIME	PROGRAMS PER DISK	CARTRIDGE REQUIRED?
ACTION REPLAY MK III SAVED WITH NORMAL TURBO	25 SECS	THREE	NO
ACTION REPLAY MK III SAVED WITH WARP 25	9.8 SECS	THREE	NO
FREEZE FRAME (MK IV)	40 SECS	TWO OR THREE	NO
FREEZE FRAME (LAZER)	25 SECS	TWO	YES
EXPERT SYSTEM	30 SECS	THREE	NO

All purchasers of Action Replay III will receive WARP 25 FREE with their cartridge. Existing Action Replay III owners can obtain WARP 25 Disk turbo by sending £2.50. post free. (No need to send your cartridge). OR obtain it FREE on the Enhancement Disk (£7.95).

### THE ACTION REPLAY ENHANCEMENT DISK

The best collection of tape to disk transfer routines for nonstandard multiloop programs (eg Dragons Lair I and II, Championship Wrestling, Summer Games, Ace of Aces, Gauntlet, Supercycle, Marble Madness, World Games). 31 titles in all. Uses our unique parameter system. No user knowledge required. Turbo load throughout. NOTE: Standard cartridge transfers normal multiloops eg Winter Games etc. etc. Disk includes file copy and disk backup utilities. PRICE £7.95 with FREE! Multicolour Slideshow for display of loading screens, hires pictures etc. saved by Action Replay. Great entertainment!

### PERFORMANCE GUARANTEE

100% Success? Rival Claims? Who's Kidding Who? Action Replay Mk III will backup any program which any other cartridge can backup - and more! It also has an unmatched range of features. Consider 'Freeze-Frame' for example, which uses more disk space, saves at slower speed, has slower tape loader, has no built in disk fast loader, no picture. Sprite or restart features and costs £10 more than Action Replay. So who's kidding who? Buy Action Replay Mk III. If you find that it does not live up to our claims return it within 7 days of receipt and your money will be refunded.

USUALLY SAMEDAY DESPATCH ON ALL ORDERS.

**datel  
electronics**

Send cheques/postal orders to:  
DATEL ELECTRONICS,  
UNIT 8/9 DEWSBURY ROAD,  
FENTON INDUSTRIAL  
ESTATE, STOKE-ON-  
TRENT. TEL: 0782 273815  
TELEX: 367257 TELSER G.



CALL 24HR CREDIT CARD LINE  
**0782 273815**

SEE OUR DOUBLE PAGE ADVERTISEMENT ELSEWHERE IN THIS MAGAZINE FOR OUR FULL RANGE OF COMMODORE ADD ONS. SEE US ON PRESTEL PAGE No 258880000A 12 PAGE CATALOGUE + ORDER PAGE.